Unsupervised Alignment of Hypergraphs with Different Scales (Online Appendix)

[If a preview does not appear properly, please download the file]

In this document, we present additional contents for the paper *"Unsupervised Alignment of Hypergraphs with Different Scales"*. These include the empirical results for Configuration 2 in which among the two input hypergraphs, the larger contains the smaller (details presented in Section 5.1 of the main manuscript). In addition, we also investigate the performance of our framework with different GAN variants and when self-loops are retained in the datasets. Furthermore, we provide the proofs for the time complexity, space complexity, and theoretical results presented in the main paper.

I. Results for Configuration 2

In this section, we present the results for Configuration 2 of the Dataset Preprocessing procedure (Section 5.1). In this setting, $\mathbf{G}_2$ contains $\mathbf{G}_1$ and the hyperedges of the original dataset that appear after those in $\mathbf{E}_1$ in chronological order.

Alignment Performance: The alignment accuracy of each method when the ratio $T_2 : T_1 \approx 2$ is in Table I. For each method, we report the average and standard deviation of 5 independent runs. The results show that HyperAlign is significantly more accurate in predicting node correspondences than other methods across all datasets.

Alignment Performance with Varying Ratios: We vary the ratio $T_2 : T_1$ and highlight the alignment accuracies of HyperAlign and the competitors in Figure 1. As the level of difference between $\mathbf{G}_1$ and $\mathbf{G}_2$ increases, the performances of all methods tend to decline, but HyperAlign consistently results in the highest alignment accuracy.

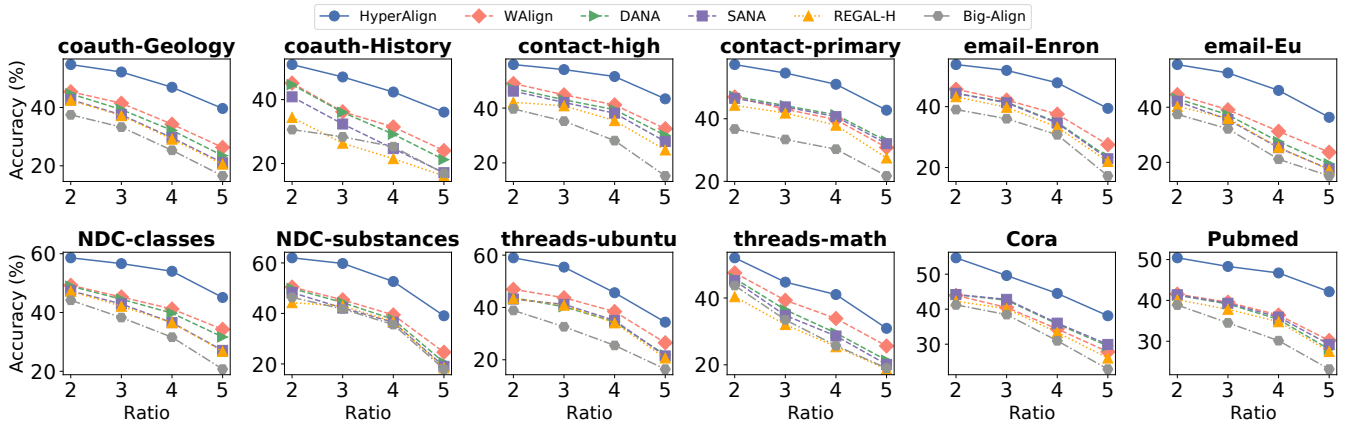

Fig. 1: Comparison in alignment accuracy of several methods when varying the difference ratio between the two hypergraphs. As the ratio increases, the alignment task becomes more difficult, and HyperAlign consistently outperforms all the competitors.

Ablation Study: Similarly to Section 5.3, we compare the alignment accuracies of HyperAlign, HA-s, HA-WC, HA-WA, and HA-WAC in Configuration 2 when $T_2 : T_1 \approx 2$. The results are summarized Table II. In each dataset, HyperAlign performs the best, followed by HA-s, HA-WC, and HA-WA, then HA-WAC.

We also vary the scale disparity ratios between two input hypergraphs and examine the performance of HyperAlign, HA-WC, HA-WA, and HA-WAC. The comparison results are demonstrated in Figure 2. As the ratio increases, the alignment decline is severe in HA-WC, HA-WA, and HA-WAC.

These results indicate that all of the three key novel modules of HyperAlign: HyperFeat- Feature Extraction, HyperCL- application of Contrastive Learning as a proxity task, and Topological Augmentation, to resolve the scale disparity, play essential roles in the alignment accuracy of HyperAlign.

TABLE I: Mean and standard deviation of alignment accuracy (in %) over five independent runs of all methods. The best alignment accuracies are in **bold**. HYPERALIGN consistently outperforms the competitors and baselines.

Dataset	SVD	HYPERFEAT	BIG-ALIGN	REGAL-G	REGAL-H	GRAD-ALIGN+	SANA	UUIL	DANA	WALIGN	HYPERALIGN
coauth-Geology	30.97 ± 0.12	36.13 ± 0.73	37.48 ± 0.51	38.65 ± 0.82	40.48 ± 0.87	40.95 ± 0.87	42.67 ± 0.94	43.27 ± 1.06	44.68 ± 1.14	45.39 ± 1.03	**54.71 ± 1.14**
coauth-History	29.47 ± 0.04	30.88 ± 0.52	30.62 ± 0.59	31.26 ± 0.74	34.41 ± 0.82	38.72 ± 0.84	40.87 ± 0.92	43.19 ± 0.96	43.86 ± 1.03	45.18 ± 1.15	**50.97 ± 1.02**
contact-high	32.18 ± 0.07	38.04 ± 0.35	39.72 ± 0.41	41.13 ± 0.57	43.32 ± 0.58	43.72 ± 0.91	46.29 ± 0.71	47.27 ± 0.71	47.13 ± 0.74	48.97 ± 0.96	**55.89 ± 0.87**
contact-primary	32.17 ± 0.09	37.04 ± 0.28	36.71 ± 0.43	43.36 ± 0.61	44.57 ± 0.51	44.76 ± 0.65	46.52 ± 0.73	46.22 ± 0.48	47.18 ± 0.75	46.95 ± 0.81	**57.32 ± 0.87**
email-Enron	32.43 ± 0.11	38.71 ± 0.46	39.04 ± 0.53	38.62 ± 0.62	43.58 ± 0.57	43.58 ± 0.67	44.39 ± 0.59	44.17 ± 0.88	44.38 ± 0.73	45.71 ± 0.87	**53.82 ± 0.91**
email-Eu	35.61 ± 0.05	39.83 ± 0.47	37.42 ± 0.51	41.67 ± 0.53	41.89 ± 0.68	42.63 ± 0.92	42.16 ± 0.71	42.37 ± 0.64	43.11 ± 0.76	44.58 ± 0.82	**55.48 ± 0.82**
NDC-classes	42.61 ± 0.07	45.23 ± 0.49	44.17 ± 0.57	46.14 ± 0.72	47.29 ± 0.84	47.58 ± 0.71	47.35 ± 0.88	47.52 ± 0.98	48.96 ± 1.05	49.26 ± 1.19	**58.61 ± 1.04**
NDC-substances	40.82 ± 0.15	46.03 ± 0.58	46.37 ± 0.52	47.58 ± 0.73	48.71 ± 0.61	47.95 ± 0.94	48.27 ± 0.83	49.32 ± 0.85	49.87 ± 0.82	50.43 ± 1.04	**62.03 ± 0.89**
threads-ubuntu	30.64 ± 0.07	35.71 ± 0.63	38.87 ± 0.86	42.18 ± 0.74	44.07 ± 0.71	42.89 ± 0.78	43.21 ± 0.87	43.81 ± 0.86	43.72 ± 1.14	46.93 ± 1.02	**58.98 ± 0.97**
threads-math	38.62 ± 0.03	42.85 ± 0.68	43.72 ± 0.74	43.59 ± 0.81	44.61 ± 0.87	44.83 ± 0.72	45.19 ± 0.91	45.29 ± 1.08	45.84 ± 1.23	47.51 ± 1.08	**52.03 ± 1.19**
Cora	36.73 ± 0.48	40.51 ± 0.67	41.23 ± 0.57	41.45 ± 0.64	42.28 ± 0.73	43.78 ± 0.87	44.19 ± 0.92	43.68 ± 1.15	44.17 ± 1.03	43.94 ± 1.12	**54.68 ± 1.03**
Pubmed	35.28 ± 0.31	37.68 ± 0.54	38.87 ± 0.71	39.68 ± 0.82	40.17 ± 0.87	40.79 ± 0.86	41.38 ± 0.94	40.94 ± 0.97	41.28 ± 1.04	41.54 ± 1.09	**50.43 ± 1.14**

TABLE II: Mean and standard deviation of alignment accuracy (in % points) over five independent runs of five versions of HYPERALIGN. The full-fledged version has the highest accuracy, demonstrating the effectiveness of the three key novel components of HYPERALIGN.

Dataset	HYPERALIGN	HA-S	HA-WC	HA-WA	HA-WAC
coauth-Geology	**54.71 ± 1.14**	51.68 ± 0.85	51.674 ± 0.97	49.13 ± 1.04	46.16 ± 0.98
coauth-History	**48.13 ± 0.91**	47.82 ± 1.03	47.82 ± 1.03	46.61 ± 0.92	45.27 ± 0.98
contact-high	**55.89 ± 0.87**	54.27 ± 0.78	53.92 ± 0.91	54.13 ± 0.95	51.03 ± 0.89
contact-primary	**57.32 ± 0.87**	54.56 ± 0.93	53.81 ± 0.87	52.81 ± 0.93	47.19 ± 0.82
email-Enron	**53.82 ± 0.91**	50.72 ± 0.89	49.52 ± 0.84	49.07 ± 0.78	46.23 ± 0.77
email-Eu	**55.48 ± 0.82**	53.87 ± 0.86	52.98 ± 0.76	49.84 ± 0.79	46.81 ± 0.92
NDC-classes	**58.61 ± 1.04**	56.53 ± 1.04	54.08 ± 0.96	53.39 ± 0.91	49.32 ± 1.13
NDC-substances	**62.03 ± 0.89**	59.62 ± 0.83	58.34 ± 0.79	59.12 ± 0.92	53.76 ± 0.96
threads-ubuntu	**58.98 ± 0.97**	57.38 ± 0.96	57.72 ± 0.89	55.67 ± 1.04	52.58 ± 1.08
threads-math	**52.03 ± 1.19**	50.27 ± 1.04	49.73 ± 0.82	49.24 ± 0.87	47.63 ± 1.08
Cora	**54.68 ± 1.03**	51.23 ± 0.96	49.37 ± 0.81	46.38 ± 0.83	42.97 ± 0.89
Pubmed	**50.43 ± 1.14**	48.76 ± 1.07	47.15 ± 0.93	45.63 ± 0.97	41.06 ± 0.92

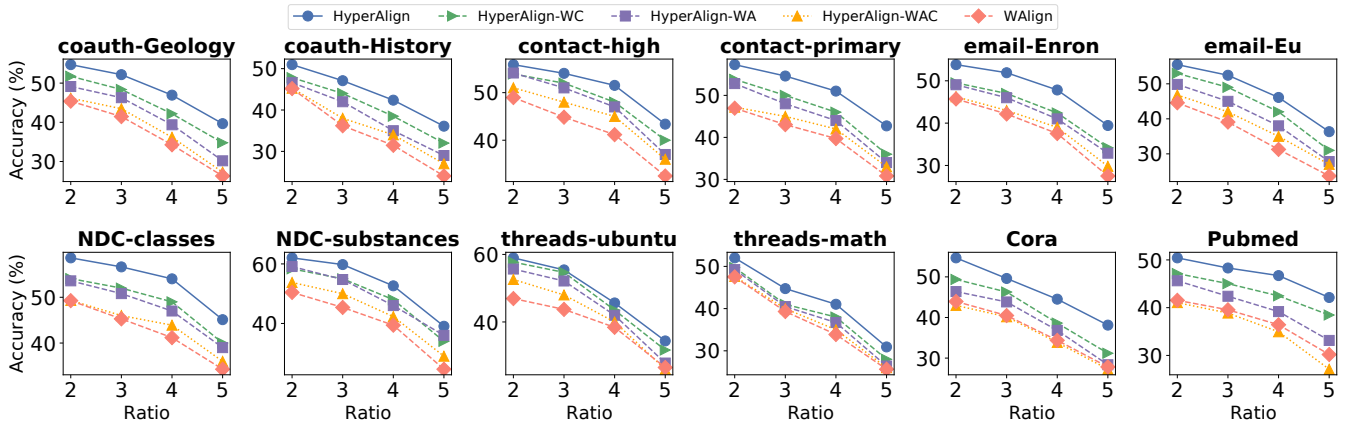

Fig. 2: Comparison in alignment accuracy of several variants of HYPERALIGN when varying the difference ratio between the two hypergraphs. The variants of HYPERALIGN consistently outperform WALIGN. As the ratio increases, the alignment task becomes more difficult, and the performance decline is severe in the variants, showing the contributions of the Contrastive Learning and Topological Augmentation modules in HYPERALIGN.

II. ADDITIONAL RESULTS

In this section, we present additional experimental results when our framework employs other GAN variants (Section II-A) and when the self-loops are retained in the datasets (Section II-B).

A. Comparison of GAN versions

In this section, we compare the performance of applying different GAN architectures to **Step 3** of HYPERALIGN. In addition to Wasserstein GAN, we also consider GAN [1] and Cycle GAN [2]. The comparison, for Configuration 1, is summarized in Table III. As the results indicate, the performances Wasserstein GAN and Cycle GAN are not much different, and both are slightly superior to that of GAN. Therefore, we only consider Wasserstein GAN for HYPERALIGN in our paper.

TABLE III: Means and standard deviations of alignment accuracy (in %) over five independent runs of all variants of HYPERALIGN with different GAN models. The best alignment accuracies are in **bold**. While Wasserstein GAN and Cycle GAN achieve comparable performances, they are slightly superior to GAN.

Dataset	GAN	CYCLEGAN	WGAN
coauth-Geology	43.21 ± 0.97	44.25 ± 1.04	**45.02 ± 1.18**
coauth-History	37.04 ± 0.94	37.25 ± 1.02	**37.94 ± 0.98**
contact-high	50.16 ± 0.72	**52.08 ± 0.83**	51.92 ± 0.74
contact-primary	51.59 ± 0.64	52.64 ± 0.72	**53.27 ± 0.65**
email-Enron	46.87 ± 0.75	47.52 ± 0.83	**48.41 ± 0.72**
email-Eu	48.65 ± 0.64	49.32 ± 0.67	**49.76 ± 0.73**
NDC-classes	50.62 ± 0.89	51.27 ± 0.82	**51.48 ± 0.87**
NDC-substances	56.93 ± 0.81	57.24 ± 0.92	**57.84 ± 0.96**
threads-ubuntu	52.31 ± 0.93	**53.48 ± 0.89**	53.37 ± 1.04
threads-math	42.17 ± 0.97	43.41 ± 1.09	**44.06 ± 1.12**
Cora	49.78 ± 0.96	50.19 ± 1.05	**50.63 ± 1.14**
Pubmed	41.27 ± 0.92	41.98 ± 1.03	**42.16 ± 1.08**

B. Effects of Self-loops

In this section, we examine the performance of HYPERALIGN when the self-loops are retained in the datasets. Note that in the main results, Table 2 in the main paper, for Configuration 1 and the results for Configuration 2, presented in Table I, the self-loops have been removed from the datasets. We compare the alignment accuracy of our method in the datasets with self-loops (HYPERALIGN-SELF) with that in the datasets where self-loops have been removed (HYPERALIGN). The results are summarized in Table IV and Table V. As the results suggest, keeping the self-loops does not result in any substantial performance difference.

TABLE IV: Comparison in performance, for Configuration 1, of our method when the self-loops are retained in the datasets (HYPERALIGN-SELF) and when the self-loops are removed (HYPERALIGN). The results suggest that recovering the self-loops in the datasets does not result in any noticeable performance difference.

Dataset	HYPERALIGN-SELF	HYPERALIGN
coauth-Geology	44.94 ± 1.07	**45.02 ± 1.18**
coauth-History	37.45 ± 1.06	**37.94 ± 0.98**
contact-high	51.38 ± 0.89	**51.92 ± 0.74**
contact-primary	52.94 ± 0.69	**53.27 ± 0.65**
email-Enron	48.02 ± 0.85	**48.41 ± 0.72**
email-Eu	49.21 ± 0.68	**49.76 ± 0.73**
NDC-classes	51.04 ± 0.79	**51.48 ± 0.87**
NDC-substances	57.31 ± 0.87	**57.84 ± 0.96**
threads-ubuntu	53.18 ± 0.93	**53.37 ± 1.04**
threads-math	43.91 ± 1.02	**44.06 ± 1.12**
Cora	50.27 ± 1.17	**50.63 ± 1.14**
Pubmed	41.88 ± 1.14	**42.16 ± 1.08**

TABLE V: Comparison in performance, for Configuration 2, of our method when the self-loops are retained in the datasets (HYPERALIGN-SELF) and when the self-loops are removed (HYPERALIGN). The results suggest that recovering the self-loops in the datasets does not result in any noticeable performance difference.

Dataset	HYPERALIGN-SELF	HYPERALIGN
coauth-Geology	53.94 ± 1.02	**54.71 ± 1.14**
coauth-History	50.15 ± 1.09	**50.97 ± 1.02**
contact-high	54.68 ± 0.82	**55.89 ± 0.87**
contact-primary	56.94 ± 0.76	**57.32 ± 0.87**
email-Enron	53.07 ± 0.82	**53.82 ± 0.91**
email-Eu	54.61 ± 0.73	**55.48 ± 0.82**
NDC-classes	58.14 ± 0.97	**58.61 ± 1.04**
NDC-substances	61.82 ± 0.84	**62.03 ± 0.89**
threads-ubuntu	58.07 ± 0.96	**58.98 ± 0.97**
threads-math	51.44 ± 1.08	**52.03 ± 1.19**
Cora	54.19 ± 1.08	**54.68 ± 1.03**
Pubmed	50.18 ± 1.05	**50.43 ± 1.14**

III. Theoretical Results and Proofs

In this section, we present the proofs for the theoretical results presented in Section 4.6 of the main paper.

A. Time Complexity of HyperAlign

The proofs for the time complexities of Steps $1-4$ of HyperAlign are presented in Lemmas 1, 2, 3, and 4, respectively. Let g_i^* denote the maximum size of a hyperedge in $\mathbf{E}_i$. We show that the time complexity of HyperAlign, when considering all of its hyperparameters as constants, is $\mathcal{O}\left(\sum_{i=1}^{2} g_i^*(|\mathbf{E}_i| + |\mathbf{V}_i|^2 \log|\mathbf{V}_i|)\right)$ in Theorem 1.

Lemma 1. For $i = 1, 2$, denote d_i as the maximum number of neighbor nodes in the k-ring neighborhood ($k = 1, 2, 3$) of a node in $\mathbf{G}_i$. Let g_i^* be the maximum size of a hyperedge in $\mathbf{E}_i$. Let l denote the length of the random walk and let w denote the size of the context windows. The time complexity of Step 1 - HyperFeat of HyperAlign is $\mathcal{O}\left(\sum_{i=1}^{2}(d_i g_i^* |\mathbf{V}_i| \log|\mathbf{V}_i| + l \log\log|\mathbf{V}_i| \log|\mathbf{V}_i|) + lw\right)$.

Proof. We construct 3 level graphs ($T = 3$ in Section 4.2), which are employed to conduct Random Walk (Section 4.2). As the Random Walk is unlikely to visit nodes having low similarities, in each level graph, we only connect each node u to only $\log|\mathbf{V}|$ nodes instead of all other nodes. They are most similar nodes with u in terms of the number of incident hyperedges. As a result, there are $|\mathbf{V}| \log|\mathbf{V}|$ edges in each level graph. To do so, we first sort all the nodes based on the number of incident hyperedges once, which takes $\mathcal{O}(|\mathbf{V}| \log|\mathbf{V}|)$ time. We then select $\frac{1}{2}\log|\mathbf{V}|$ nodes that rank higher than u and $\frac{1}{2}\log|\mathbf{V}|$ nodes that rank lower than u. If one direction (higher or lower) has fewer than $\frac{1}{2}\log|\mathbf{V}|$ nodes, we make up in the opposite direction to have enough $\log|\mathbf{V}|$ nodes to connect with u in each level graph. The total time complexity of determining all the connections for the 3 level graphs is $\mathcal{O}(|\mathbf{V}| \log|\mathbf{V}|)$.

Constructing an edge between each pair of nodes (u, v) requires computing the distance between u and v. In order to measure the distance $m(s(N_k(u)), s(N_k(v)))$ (Eq. (2) in the main paper) in each layer-k graph ($k = 1, 2, 3$), we compute the distance between two sequences $s(N_k(v))$ and $s(N_k(v))$. For each hyperedge size $g = 2, ..., g^*$ (g^*: the maximum hyperedge size in $\mathbf{E}$), we compute a distance, denoted as $m_g(s(N_k(u)), s(N_k(v)))$, by only considering the number of size-g incident hyperedges $\lambda_w(g)$ of each node w in $N_k(u)$ and $N_k(v)$. In particular, we compute the Dynamic Time Warping [3] between two respective sequences of $\{\lambda_w(g)\}_{w \in s(N_k(u))}$ and $\{\lambda_{w'}(g)\}_{w' \in s(N_k(v))}$ of $s(N_k(u))$ and $s(N_k(v))$. After that, we obtain the distance $m(s(N_k(u)), s(N_k(v)))$ by taking the summation $m(s(N_k(u)), s(N_k(v))) = \sum_{g=2}^{g^*} m_g(s(N_k(u)), s(N_k(v)))$.

For each hypergraph $\mathbf{G}_i$, g_i^* is the maximum hyperedge size in $\mathbf{E}_i$. Computing the distance between 2 nodes at each level now requires matching $(g_i^* - 1)$ pairs of sequences in which each sequence of a list of integers. The time complexity to match 1 pair of sequences is $\mathcal{O}(\max\{|N_k(u)|, |N_k(v)|\}) = \mathcal{O}(d_i)$ [3], so the time to compute the distance between each pair of nodes in each level graph (match $(g_i^* - 1)$ pairs of sequences) is $\mathcal{O}(g_i^* d_i)$. As there are $|\mathbf{V}_i| \log|\mathbf{V}_i|$ edges in each level graph, the total complexity of constructing 3 level graphs for hypergraph $\mathbf{G}_i$ is $\mathcal{O}(d_i g_i^* |\mathbf{V}_i| \log|\mathbf{V}_i|)$.

Each step of the random walk requires choosing the next node from $\log|\mathbf{V}_i|$ nodes, which takes $\mathcal{O}(\log\log|\mathbf{V}_i| \log|\mathbf{V}_i|)$ time [4]. Therefore, the total time complexity to build the corpus from the random walk is $\mathcal{O}(l \log\log|\mathbf{V}_i| \log|\mathbf{V}_i|)$.

In addition, the time to iterate through the sequence of nodes, obtained by random walk, and build contexts is $\mathcal{O}(lw)$

Thus, the time complexity of Step 1 of HyperAlign- HyperFeat is: $\mathcal{O}\left(\sum_{i=1}^{2}(d_i g_i^* |\mathbf{V}_i| \log|\mathbf{V}_i| + l \log\log|\mathbf{V}_i| \log|\mathbf{V}_i|) + lw\right)$. □

Lemma 2. Let F denote the node embedding dimension and L denote the number of layers in the hypergraph neural network Φ. Assume that there are Γ_{cl} Contrastive Learning iterations in Step 2. The time complexity of Step 2 - HyperCL of HyperAlign is $\mathcal{O}\left(\Gamma_{\text{cl}} \sum_{i=1}^{2}(LF \sum_{e \in \mathbf{E}_i} |e| + F|\mathbf{V}_i|^2)\right)$.

Proof. The forward propagation phase of each iteration in Step 2 consists of Corruption, Message Passing, and Loss Computation. The time complexity of each component is as follows:

Corruption: this step requires iterating through all entries in the node feature matrix and all member nodes of all hyperedges, which requires $\mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i| \times F + \sum_{e \in \mathbf{E}_i} |e|)\right)$.

Message Passing: the message passing scheme (Eq. (1)) in each layer requires iterating through all F-dimension member node embeddings to obtain hyperedge embeddings and iterating through all incident hyperedge embeddings to obtain the node embeddings. Therefore, the total time complexity of message passing in L layers is: $\mathcal{O}\left(LF \sum_{i=1}^{2} \sum_{e \in \mathbf{E}_i} |e|\right)$.

Loss Computation: the computation of each node-level loss (Eq. (4)) requires iterating through all F-dimension node embeddings. Thus, the time complexity to compute Eq. (5) is: $\mathcal{O}\left(F(|\mathbf{V}_1|^2 + |\mathbf{V}_2|^2)\right)$.

Therefore, the time complexity of each iteration in Step 2 is: $\mathcal{O}\left(\sum_{i=1}^{2}(LF \sum_{e \in \mathbf{E}_i} |e| + F|\mathbf{V}_i|^2)\right)$. As there are Γ_{cl} iterations, the time complexity of Step 2 is: $\mathcal{O}\left(\Gamma_{\text{cl}} \sum_{i=1}^{2}(LF \sum_{e \in \mathbf{E}_i} |e| + F|\mathbf{V}_i|^2)\right)$. □

Lemma 3. Let F denote the node embedding dimension and L denote the number of layers in the hypergraph neural network Φ. Assume that there are Γ_{g} iterations of GAN training in Step 3. Assume that in each iteration of GAN training, there are Γ_{d} iterators to train the discriminator f_θ and Γ_{r} iterations to train the reconstruction functions $\mathbf{r}_1$ and $\mathbf{r}_2$. The time complexity of Step 3 is $\mathcal{O}\left(\Gamma_{\mathrm{g}}\left[(F+t)|\mathbf{V}_1||\mathbf{V}_2| + \sum_{i=1}^{2} F(\Gamma_{\mathrm{d}}+\Gamma_{\mathrm{r}})|\mathbf{V}_i| + (LF+t)\sum_{e\in\mathbf{E}_i}|e|\right]\right)$.

Proof. In Step 3, there are Γ_{g} iterations of GAN training. Each iteration consists of Topological Augmentation, Γ_{d} iterations of Discriminator Loss Computation, Γ_{r} iterations of Reconstruction Loss Computation, Message Passing, and Computation of Generator Loss. The time complexity of each component is as follows:

Topological Augmentation: this step requires computing all pair-wise similarities between the embedding of each node in V_1 and that of each node in V_2, each of which $\in R^F$. We then retrieve t most similar nodes for each node and compute t soft memberships for each original node's membership in each hyperedge. The total time complexity is: $\mathcal{O}\left((F+t)|\mathbf{V}_1||\mathbf{V}_2| + t\sum_{i=1}^{2}\sum_{e\in\mathbf{E}_i}|e|\right)$.

Discriminator Loss: the time complexity of computing the discriminator loss (Eq. (13)) in Γ_{d} iterations is $\mathcal{O}\left(\Gamma_{\mathrm{d}}F(|\mathbf{V}_1|+|\mathbf{V}_2|)\right)$.

Reconstruction Loss: the time complexity of computing the reconstruction loss (Eq. (7)) in Γ_{r} iterations is $\mathcal{O}\left(\Gamma_{\mathrm{r}}F(|\mathbf{V}_1|+|\mathbf{V}_2|)\right)$.

Message Passing: according to the proof of Lemma 2, the time complexity of this component is $\mathcal{O}\left(LF\sum_{i=1}^{2}(\sum_{e\in\mathbf{E}_i}|e|)\right)$

Generator Loss: the time complexity for computing the generator loss (Eq. (8)) is $\mathcal{O}\left(F(|\mathbf{V}_1|+|\mathbf{V}_2|)\right)$.

The time complexity of each GAN training iteration in Step 3 is $\mathcal{O}\left((F+t)|\mathbf{V}_1||\mathbf{V}_2| + \sum_{i=1}^{2} F(\Gamma_{\mathrm{d}}+\Gamma_{\mathrm{r}})|\mathbf{V}_i| + (LF+t)\sum_{e\in\mathbf{E}_i}|e|\right)$.

Therefore, the total time complexity of Step 3 is $\mathcal{O}\left(\Gamma_{\mathrm{g}}\left[(F+t)|\mathbf{V}_1||\mathbf{V}_2| + \sum_{i=1}^{2} F(\Gamma_{\mathrm{d}}+\Gamma_{\mathrm{r}})|\mathbf{V}_i| + (LF+t)\sum_{e\in\mathbf{E}_i}|e|\right]\right)$. □

Lemma 4. Let F denote the node embedding dimension. The time complexity of Step 4 - Alignment Prediction is $\mathcal{O}\left(F|\mathbf{V}_1||\mathbf{V}_2|\right)$.

Proof. First, HYPERALIGN computes the pairwise similarities in node embeddings between each node in $\mathbf{V}_1$ and each node in $\mathbf{V}_2$, which takes $\mathcal{O}\left(F|\mathbf{V}_1||\mathbf{V}_2|\right)$ time.

This step also involves iterating through each node v in $\mathbf{G}_1$ and finds the node in $\mathbf{G}_2$ having the most similar embedding to that of v and thus has time complexity $O(|\mathbf{V}_1||\mathbf{V}_2|)$.

Therefore, the total time complexity of Step 4 is $\mathcal{O}\left(F|\mathbf{V}_1||\mathbf{V}_2|\right)$. □

Based on the time complexity of each step, we now derive the time complexity of HYPERALIGN when the hyperparameters are regarded as constants.

Theorem 1 (TIME COMPLEXITY OF HYPERALIGN)**.** For $i=1,2$, denote d_i as the maximum number of neighbor nodes in the k-ring neighborhood ($k=1,2,3$) of a node in $\mathbf{G}_i$. Let g_i^* be the maximum size of a hyperedge in $\mathbf{E}_i$. Let l denote the length of the random walk and let w denote the size of the context windows. Let F denote the node embedding dimension and L denote the number of layers in the hypergraph neural network Φ. Assume that there are Γ_{cl} Contrastive Learning iterations in Step 2 and Γ_{g} iterations of GAN training in Step 3. Assume that in each iteration of GAN training, there are Γ_{d} iterators to train the discriminator f_θ and Γ_{r} iterations to train the reconstruction functions $\mathbf{r}_1$ and $\mathbf{r}_2$. The time complexity for each step of HYPERALIGN is as follows:

- Step 1 - HYPERFEAT: $\mathcal{O}\left(\sum_{i=1}^{2} d_i g_i^*(|\mathbf{V}_i|\log|\mathbf{V}_i| + l\log\log|\mathbf{V}_i|\log|\mathbf{V}_i|) + lw\right)$
- Step 2 - HYPERCL: $\mathcal{O}\left(\Gamma_{\mathrm{cl}}\sum_{i=1}^{2}(LF\sum_{e\in\mathbf{E}_i}|e| + F|\mathbf{V}_i|^2)\right)$
- Step 3 - HYPERAUG: $\mathcal{O}\left(\Gamma_{\mathrm{g}}\left[(F+t)|\mathbf{V}_1||\mathbf{V}_2| + \sum_{i=1}^{2} F(\Gamma_{\mathrm{d}}+\Gamma_{\mathrm{r}})|\mathbf{V}_i| + (LF+t)\sum_{e\in\mathbf{E}_i}|e|\right]\right)$
- Step 4 - Alignment Prediction: $\mathcal{O}\left(F|\mathbf{V}_1||\mathbf{V}_2|\right)$

Also, when considering all hyperparameters of HYPERALIGN as constants, the time complexity becomes $\mathcal{O}\left(\sum_{i=1}^{2} g_i^*(|\mathbf{E}_i| + |\mathbf{V}_i|^2\log|\mathbf{V}_i|)\right)$.

Proof. Applying the results of Lemmas 1, 2, 3, and 4, we prove the time complexity of each step of HYPERALIGN.

For Step 1, as $d_i = \mathcal{O}\left(|\mathbf{V}|_i\right)$ and $\log\log|\mathbf{V}_i|\log|\mathbf{V}_i|$ is dominated by $|\mathbf{V}_i|\log|\mathbf{V}_i|$, the time complexity of Step 1 is $\mathcal{O}\left(\sum_{i=1}^{2} g_i^*|\mathbf{V}_i|^2\log|\mathbf{V}_i|\right)$, considering that l and w are hyperparameters.

In Step 2, we have that $\sum_{e\in\mathbf{E}_i}|e| = \mathcal{O}\left(g_i^*|\mathbf{E}_i|\right)$. When considering Γ_{cl}, L, and F as constants with respect to HYPERALIGN, the time complexity of this step becomes $\mathcal{O}\left(\sum_{i=1}^{2}(g_i^*|\mathbf{E}_i| + |\mathbf{V}_i|^2)\right)$.

Similarly, for Step 3 - HYPERAUG, assuming that all the hyperparameters of HYPERALIGN are constants, the time complexity is $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2| + \sum_{i=1}^{2}|\mathbf{V}_i| + \sum_{e\in\mathbf{E}_i}|e|\right) = \mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i|^2 + g_i^*|\mathbf{E}_i|)\right)$.

Treating the hyperparameter F as a constant, the time complexity of Step 4 is $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2|\right) = \mathcal{O}\left(|\mathbf{V}_1|^2 + |\mathbf{V}_2|^2\right)$.

Summing the time complexities of all steps, the time complexity of HYPERALIGN, when considering all hyperparameters as constants, becomes $\mathcal{O}\left(\sum_{i=1}^{2} g_i^*(|\mathbf{E}_i| + |\mathbf{V}_i|^2 \log|\mathbf{V}_i|)\right)$.

□

B. SPACE COMPLEXITY OF HYPERALIGN

In this section, we derive the space complexity of HYPERALIGN. The proofs for the time complexities of Steps $1-4$ of HYPERALIGN are presented in Lemmas 5, 6, 7, and 8, respectively. We show that when considering all the hyperparameters as constants, the space complexity of HYPERALIGN is $\mathcal{O}\left(\sum_{i=1}^{2}(\sum_{e\in\mathbf{E}_i}|e| + |\mathbf{V}_i|\log|\mathbf{V}_i|) + |\mathbf{V}_1||\mathbf{V}_2|\right)$.

Lemma 5. Let l denote the length of the random walk in Step 1 and w denote the size of the context windows. Denote F as the node feature dimension. The space complexity of Step 1 - HYPERFEAT of HYPERALIGN is $\mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i|\log|\mathbf{V}_i| + F|\mathbf{V}_i|) + lw\right)$.

Proof. The random walk in Step 1 requires storing the transition probabilities that correspond to $|\mathbf{V}_i|\log|\mathbf{V}_i|$ edges in a level graph for each input hypergraph. This requires $\mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i|\log|\mathbf{V}_i|\right)$ space.

Moreover, storing the corpus, the result of the random walk, and contexts requires $\mathcal{O}\left(lw\right)$ space.

As this step outputs node features, storing these node features requires $\mathcal{O}\left(\sum_{i=1}^{2}(F|\mathbf{V}_i|\right)$ space.

Therefore, the total space complexity for Step 1 is: $\mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i|\log|\mathbf{V}_i| + F|\mathbf{V}_i|) + lw\right)$ □

Lemma 6. Let F denote the node embedding dimension and L denote the number of layers in the hypergraph neural network Φ. The space complexity of Step 2 - HYPERCL of HYPERALIGN is $\mathcal{O}\left(\sum_{i=1}^{2}[\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)]\right)$.

Proof. The forward propagation phase of each iteration in Step 2 consists of Corruption, Message Passing, and Loss Computation. The space complexity of each component is as follows:

Corruption: this step requires creating corrupted node feature matrices and masked hyperedges, which requires a space complexity of: $\mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i| \times F + \sum_{e\in\mathbf{E}_i}|e|)\right)$.

Message Passing: the message passing scheme (Eq. (1)) requires storing the incidence relationship between nodes and hyperedges, which requires an $\mathcal{O}\left(\sum_{i=1}^{2}(\sum_{e\in\mathbf{E}_i}|e|)\right)$ space. In addition, each layer of the hypergraph neural network requires storing all F-dimension node embeddings and hyperedge embeddings, which occupies an $\mathcal{O}\left(\sum_{i=1}^{2} LF(|\mathbf{V}_i| + |\mathbf{E}_i|)\right)$space. Therefore, the total time complexity of message passing in L layers is: $\mathcal{O}\left(\sum_{i=1}^{2}[\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)]\right)$.

Loss Computation: the computation of each node-level loss (Eq. (3)) requires storing all F-dimension node embeddings. Therefore, the space complexity to compute Eq.(4) is $\mathcal{O}\left(F(|\mathbf{V}_1| + |\mathbf{V}_2|)\right)$.

Therefore, the total space complexity of Step 2 is $\mathcal{O}\left(\sum_{i=1}^{2}[\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)]\right)$. □

Lemma 7. Let F denote the node embedding dimension and L denote the number of layers in the hypergraph neural network Φ. The space complexity of Step 3 is $\mathcal{O}\left(\sum_{i=1}^{2}[t\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)] + |\mathbf{V}_1||\mathbf{V}_2|\right)$.

Proof. Step 3 involves Topological Augmentation, Discriminator Loss Computation, Reconstruction Loss Computation, Message Passing, and Computation of Generator Loss. The space complexity of each component is as follows:

Topological Augmentation: this step requires storing all node embeddings and pair-wise similarities between the embedding of each node in V_1 and that of each node in V_2, which takes an $\mathcal{O}\left(F(|\mathbf{V}_1| + |\mathbf{V}_2|) + |\mathbf{V}_1||\mathbf{V}_2|\right)$ space. HYPERALIGN then augments hyperedges by which each augmented hyperedge may have up to t times more member nodes than its corresponding original hyperedge. The space complexity for such augmentation is: $\mathcal{O}\left(t\sum_{i=1}^{2}\sum_{e\in\mathbf{E}_i}|e|\right)$. Thus, the total space complexity for Topological Augmetation is $\mathcal{O}\left(F(|\mathbf{V}_1| + |\mathbf{V}_2|) + |\mathbf{V}_1||\mathbf{V}_2| + t\sum_{i=1}^{2}\sum_{e\in\mathbf{E}_i}|e|\right)$.

Message Passing: according to the proof of Lemma 6, the space complexity of this component is $\mathcal{O}\left(\sum_{i=1}^{2}[\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)]\right)$.

Loss Computation: computing any loss (Eq. (12), EQ. (7), Eq. (8)) requires storing all node embeddings, which takes a space complexity of $\mathcal{O}\left(F(|\mathbf{V}_1| + |\mathbf{V}_2|)\right)$.

Therefore, the total space complexity of Step 3 is $\mathcal{O}\left(\sum_{i=1}^{2}[t\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)] + |\mathbf{V}_1||\mathbf{V}_2|\right)$. □

Lemma 8. Let F abe the node embedding dimension. The space complexity of Step 4 - Alignment Prediction is $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2| + F(|\mathbf{V}_1| + |\mathbf{V}_2|)\right)$.

Proof. In this step, we need to store the embeddings of all nodes in the two input hypergraphs, which requires a space complexity of $\mathcal{O}\left(F(|\mathbf{V}_1| + |\mathbf{V}_2|)\right)$.

Moreover, HYPERALIGN in this step computes the pairwise similarities in node embedding between every node in $\mathbf{V}_1$ and every node in $\mathbf{V}_2$, taking an $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2|\right)$ space.

Thus, the space complexity of Step 4 is $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2| + F(|\mathbf{V}_1| + |\mathbf{V}_2|)\right)$. □

Given the space complexities of all steps of HYPERALIGN, we now derive the space complexity of our method when all the hyperparameters are considered constants.

Theorem 2 (SPACE COMPLEXITY OF HYPERALIGN)**.** Let l be the length of the random walk in Step 1 and w be the size of the context windows. Let F denote the node embedding dimension and L denote the number of layers in the hypergraph neural network Φ. The space complexity for each step of HYPERALIGN is as follows:

- Step 1 - HYPERFEAT: $\mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i|\log|\mathbf{V}_i| + F|\mathbf{V}_i|) + lw\right)$
- Step 2 - HYPERCL: $\mathcal{O}\left(\sum_{i=1}^{2}[\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)]\right)$
- Step 3 - HYPERAUG: $\mathcal{O}\left(\sum_{i=1}^{2}[t\sum_{e\in\mathbf{E}_i}|e| + LF(|\mathbf{V}_i| + |\mathbf{E}_i|)] + |\mathbf{V}_1||\mathbf{V}_2|\right)$
- Step 4 - Alignment Prediction: $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2| + F(|\mathbf{V}_1| + |\mathbf{V}_2|)\right)$

Also, when considering all hyperparameters of HYPERALIGN as constants, the time complexity becomes: $\mathcal{O}\left(\sum_{i=1}^{2}(\sum_{e\in\mathbf{E}_i}|e| + |\mathbf{V}_i|\log|\mathbf{V}_i|) + |\mathbf{V}_1||\mathbf{V}_2|\right)$.

Proof. Applying the results of Lemmas 5, 6, 7, 4, we derive the space complexity of each step.

For Step 1, as $|\mathbf{V}_i|$ is dominated by $|\mathbf{V}_i|\log|\mathbf{V}_i|$, the space complexity of Step 1 becomes $\mathcal{O}\left(\sum_{i=1}^{2}(|\mathbf{V}_i|\log|\mathbf{V}_i|)\right)$ when considering F, l, w as constants.

Similarly, treating L and F as constants, as $|\mathbf{E}_i|$ is $\mathcal{O}\left(\sum_{e\in\mathbf{E}_i}|e|\right)$, the space complexity of Step 2 becomes $\mathcal{O}\left(\sum_{i=1}^{2}(\sum_{e\in\mathbf{E}_i}|e| + |\mathbf{V}_i|)\right)$.

For Step 3, we have that $|\mathbf{E}_i|$ is $\mathcal{O}\left(\sum_{e\in\mathbf{E}_i}|e|\right)$ and $(|\mathbf{V}_1| + |\mathbf{V}_2|)$ is dominated by $|\mathbf{V}_1||\mathbf{V}_2|$. As a result, when the hyperparameters are regarded as constants, the space complexity of Step 3 is $\mathcal{O}\left(\sum_{i=1}^{2}\sum_{e\in\mathbf{E}_i}|e| + |\mathbf{V}_1||\mathbf{V}_2|\right)$.

Finally, since $(|\mathbf{V}_1| + |\mathbf{V}_2|)$ is $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2|\right)$, Step 4 requires a space complexity of $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2|\right)$ when F is considered constant.

Summing the space complexities of all steps and noting that $(|\mathbf{V}_1| + |\mathbf{V}_2|)$ is $\mathcal{O}\left(|\mathbf{V}_1||\mathbf{V}_2|\right)$, the space complexity of HYPERALIGN, when all hyperparameters are considered constants, is $\mathcal{O}\left(\sum_{i=1}^{2}(\sum_{e\in\mathbf{E}_i}|e| + |\mathbf{V}_i|\log|\mathbf{V}_i|) + |\mathbf{V}_1||\mathbf{V}_2|\right)$. □

C. HYPERFEAT AS MATRIX FACTORIZATION

In a special case, HYPERFEAT only considers the structural differences among nodes at a particular l-hop neighborhood and generates a sequence of nodes by Random Walk on the l-level graph $G^{[l]}$. In such a case, we show that when the corpus length L is sufficiently large, HYPERFEAT obtains the node features by performing an implicit factorization of a constant matrix.

The proof for this property, Theorem 3, follows the ideas of [5] by showing that HYPERFEAT performs an implicit factorization of a matrix M, which is uniquely defined for HYPERFEAT, and that M converges to a constant matrix $\mathbf{M}$ when $L \to \infty$.

In $G^{[l]}$, the Random Walk (RW) determines the next node in the sequence from the current node u as follows:

- with probability $(1-q) > 0$, the next node in the sequence is u
- for any other node v, the probability that v is the next node in the sequence is equal to: $q\frac{e^{-d_l(n,v)}}{\sum_{t\in\mathbf{V},t!=u}e^{-d_l(n,t)}} > 0$.

As the Random Walk decides the next node in the sequence solely based on the current node, the process can be modeled as a Markov Chain $\mathcal{M}$ in which the set $\mathbb{S}$ of states is the set of nodes $\mathbf{V}$ and the transition probability $\mathbf{P}$ indicates the transition probabilities. The entry $\mathbf{P}_{u,v}$ is defined for each node pair (u, v) as above: $\mathbf{P}_{u,v} = 1-q$ if $u = v$ and $\mathbf{P}_{u,v} = q\frac{e^{-d_l(n,v)}}{\sum_{t\in\mathbf{V},t!=u}e^{-d_l(n,t)}}$ if $u! = v$.

The proofs make use of the following notations:

- $\mathbf{P}_{w,c}$: the entry in the row corresponding to node w and the column corresponding to node c in the transition probability matrix $\mathbf{P}$.
- $\mathbf{P}^r$: the exponentiation of $\mathbf{P}$ with the exponent r.
- $(\mathbf{P}^r)_{w,c}$: the entry in the row corresponding to node w and the column corresponding to node c in matrix $\mathbf{P}^r$.

We first show that $\mathcal{M}$ converges to a unique stationary distribution π.

Lemma 9. The Markov Chain $\mathcal{M}$ converges to a unique stationary distribution π.

Proof. As $\mathbf{P}_{u,v} > 0 \, \forall u, v \in \mathbf{V}$ and $\mathbb{S} = \mathbf{V}$, $\mathbf{P}_{u,v} > 0 \, \forall u, v \in \mathbb{S}$ the Markov Chain $\mathcal{M}$ is ergodic.

As $\mathcal{M}$ is ergodic and has a finite number of states (as $|\mathbf{V}|$ is finite), $\mathcal{M}$ converges to a stationary distribution π. □

Denote π_u the probability corresponding to node u in the stationary distribution π.

Upon obtaining the sequence of nodes by the Random Walk, HYPERFEAT achieves the node features by applying Skip-Gram with Negative Sampling (SGNS). For a language model, from the corpus, for each value $r = 1, ..., \mathcal{W}$ ($\mathcal{W}$ is the maximum windows size to generate contexts), SGNS constructs $\mathcal{D}_{\overrightarrow{r}}$ as the list of all (word, context) pairs that appear exactly r time-steps away in the sequence (by Random Walk) and $\mathcal{D}_{\overleftarrow{r}}$ as the list of all (context, word) pairs that appear exactly r time-steps away in the sequence.

Similarly, in HYPERFEAT, SGNS constructs $\mathcal{D}_{\overrightarrow{r}}$ as the list of all (word node, context node) pairs that appear exactly r time-steps away in the corpus and $\mathcal{D}_{\overleftarrow{r}}$ as the list of all (context node, word node) pairs that appear exactly r time-steps away in the corpus. Note that each context node or word node is a node in $\mathbf{V}$ (or a state in the state space $\mathbb{S}$ of the Markov Chain $\mathcal{M}$).

Denote $n(w,c)_{\overrightarrow{r}}$ as the number of appearances of pair (w,c) in $\mathcal{D}_{\overrightarrow{r}}$ and $n(c,w)_{\overleftarrow{r}}$ as the number of appearances of pair (c,w) in $\mathcal{D}_{\overleftarrow{r}}$. Next, we show that for each pair (w,c) of word node w and context node c, we have $\frac{n(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} \to \pi_c(\mathbf{P}^r)_{w,c}$ and $\frac{n(c,w)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \to \pi_c(\mathbf{P}^r)_{c,w}$ as $L \to \infty$.

Before going into the proof, we cite the following lemma (without repeating its proof) regarding the law of large numbers for a random sequence [6].

Lemma 10 (S.N. Bernstein Law of Large Numbers)**.** Let $y_1, y_2, ...$ be a sequence of random variables with a finite expectation $\mathbb{E}[y]$ and finite variance. Assume the covariances satisfy that $\text{Cov}(y_i, y_j) \to 0$ as $|j-i| \to \infty$. Then the law of large numbers holds for the sequence: $\frac{1}{L}\sum_{i=1}^{L} y_i \to \mathbb{E}[y]$ as $L \to \infty$.

Applying Lemma 10, we prove the convergence of the terms $\frac{n(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|}$ and $\frac{n(c,w)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \to \pi_c$ in Lemma 11.

Lemma 11. $\frac{n(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} \to \pi_w(\mathbf{P}^r)_{w,c}$ and $\frac{n(c,w)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \to \pi_c(\mathbf{P}^r)_{c,w}$ as $L \to \infty$

Proof. Denote v_i as the node in the i-th time step of the node sequence generated by the RW. Denote $y_j, j = 1, ..., L - \mathcal{W}$, as the indicator function for the event that $v_j = w$ and $v_{j+r} = c$.

We then have: $\frac{n(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} = \frac{1}{L-\mathcal{W}}\sum_{j=1}^{L-\mathcal{W}} y_j$. We also have $\mathbb{E}(y_j) = \pi_w(\mathbf{P}^r)_{w,c}$.

Consider $\mathbb{E}(y_i y_j)$ with $j - i > r$: $\mathbb{E}(y_i y_j) = \text{Prob}(v_i = w, v_{i+r} = c, v_j = w, v_{j+r} = c) = \pi_w(\mathbf{P}^r)_{w,c}(\mathbf{P}^{j-i-r})_{c,w}(\mathbf{P}^r)_{w,c}$.

Considering the variance of y_i and y_j: $\text{Cov}(y_i, y_j) = \mathbb{E}(y_i y_j) - \mathbb{E}(y_i)\mathbb{E}(y_j) = \pi_w(\mathbf{P}^r)_{w,c}\Big((\mathbf{P}^{j-i-r})_{c,w} - \pi_w\Big)(\mathbf{P}^r)_{w,c}$.

As $\mathcal{M}$ converges to the stationary distribution π (Lemma 9), we have: $(\mathbf{P}^{j-i-r})_{c,w} - \pi_w \to 0$ as $(j-i) \to \infty$. (Note that r is a constant with respect to HYPERFEAT). As a result, $\text{Cov}(y_i, y_j) \to 0$ as $(j-i) \to \infty$.

Therefore, by Lemma 10, $\frac{n(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} = \frac{1}{L-\mathcal{W}}\sum_{j=1}^{L-\mathcal{W}} y_j \to \mathbb{E}(y_1) = \pi_w(\mathbf{P}^r)_{w,c}$ as $L \to \infty$.

Similarly, we can also show that: $\frac{n(c,w)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \to \pi_c(\mathbf{P}^r)_{c,w}$ as $L \to \infty$. □

Let $\mathcal{D}$ be the concatenation of all $\mathcal{D}_{\overrightarrow{r}}$ and $\mathcal{D}_{\overleftarrow{r}}$ (for $r = 1, ..., \mathcal{W}$). Note that for each sub-sequence of size $\mathcal{W}$: $v_{k\mathcal{W}+1}, ..., v_{k\mathcal{W}+\mathcal{W}}$ ($k \in \mathbb{N}$), we only add $(v_{k\mathcal{W}+1}, v_{k\mathcal{W}+r})$ to $\mathcal{D}_{\overrightarrow{r}}$ and $(v_{k\mathcal{W}+r}, v_{k\mathcal{W}+1})$ to $\mathcal{D}_{\overleftarrow{r}}$. As a result $\frac{|\mathcal{D}_{\overrightarrow{r}}|}{|\mathcal{D}|} = \frac{|\mathcal{D}_{\overleftarrow{r}}|}{|\mathcal{D}|} = \frac{1}{2\mathcal{W}}$.

Denote $n(w,c)$ as the number of appearances of the word node-context node pair (w,c) in $\mathcal{D}$. We next show that: $\frac{n(w,c)}{|\mathcal{D}|} \to \frac{1}{2\mathcal{W}}\sum_{r=1}^{\mathcal{W}}(\pi_w(\mathbf{P}^r)_{w,c} + \pi_c(\mathbf{P}^r)_{c,w})$ as $L \to \infty$.

Lemma 12. When $L \to \infty$, $\frac{n(w,c)}{|\mathcal{D}|} \to \frac{1}{2\mathcal{W}}\sum_{r=1}^{\mathcal{W}}(\pi_w(\mathbf{P}^r)_{w,c} + \pi_c(\mathbf{P}^r)_{c,w})$

Proof. We have:

$$\begin{aligned}
\frac{n(w,c)}{|\mathcal{D}|} &= \frac{\sum_{r=1}^{\mathcal{W}}(n(w,c)_{\overrightarrow{r}} + n(w,c)_{\overleftarrow{r}})}{\sum_{r=1}^{\mathcal{W}}(|\mathcal{D}_{\overrightarrow{r}}| + |\mathcal{D}_{\overleftarrow{r}}|)} \\
&= \frac{1}{2\mathcal{W}}\sum_{r=1}^{\mathcal{W}}\Big(\frac{n(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} + \frac{n(c,w)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|}\Big) \\
&\to \frac{1}{2\mathcal{W}}\sum_{r=1}^{\mathcal{W}}(\pi_w(\mathbf{P}^r)_{w,c} + \pi_c(\mathbf{P}^r)_{c,w}),
\end{aligned} \tag{1}$$

where the last claim is based on Lemma 11. □

Denote $n(w)$ as the number of appearances of node w in the list $\mathcal{D}$. We now show that $\frac{n(w)}{|\mathcal{D}|}$ converges to a constant when $L \to \infty$.

Lemma 13. $\frac{n(w)}{|\mathcal{D}|}$ converges to a constant when $L \to \infty$.

Proof. As $\frac{n(w)}{|\mathcal{D}|}$ can be obtained by marginalizing $\frac{n(w,c)}{|\mathcal{D}|}$ with respect to c (by a finite sum), we derive that that $\frac{n(w)}{|\mathcal{D}|}$ also converges to a constant (with respect to w) as $L \to \infty$. □

We now prove Theorem 3 for the matrix factorization property of HYPERALIGN.

Theorem 3 (HYPERFEAT AS MATRIX FACTORIZATION)**.** Assume that HYPERFEAT only generates the corpus by Random Walk on one graph $G^{[l]}$. When the corpus length $L \to \infty$, the embeddings produced by HYPERFEAT result from an implicit factorization of a constant matrix $\mathbf{M}^*$.

Proof. Let $\mathbf{V}$ denote the set of nodes of the hypergraph that we apply HYPERFEAT to extract node features.

According to [7], performing SGNS to extract the node feature vectors with b negative samples is equivalent to implicitly factorizing a $|\mathbf{V}| \times |\mathbf{V}|$ matrix M in which the entry in row corresponding to node w and column corresponding to node c is equal to $M[w][c] = \log \frac{n(w,c)/|\mathcal{D}|}{(n(w)/|\mathcal{D}|)(n(c)/|\mathcal{D}|)} - \log b$.

According to Lemma 12 and Lemma 13, each term in $M[w][c]$ converges to a constant when $L \to \infty$. Therefore, $M[w][c]$ converges to a constant when $L \to \infty$. As a result, M converges to a constant matrix $\mathbf{M}^*$ as $L \to \infty$. □

D. HYPERFEAT *as an Isomorphism Test*

One common approach when extracting node features based on matrix factorization is to conduct Singular Value Decomposition (SVD) on the matrix [5], [7], [8]. Let HYPERFEAT* be a version of HYPERFEAT that extracts node feature matrix $\mathbf{X}$ by performing Singular Value Decomposition on $\mathbf{M}^* = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}, \mathbf{X} = \mathbf{U}\sqrt{\mathbf{\Sigma}}$.

We first show that HYPERFEAT* is invariant up to any node permutation. As a corollary, we show that HYPERFEAT* constitutes an Isomorphism Test for hypergraphs.

Lemma 14. Suppose that hypergraph $\mathbf{G}_2$ is obtained by applying a node permutation σ, corresponding to the permutation matrix Q, on hypergraph $\mathbf{G}_1$. Suppose that HYPERFEAT* obtains the node features by factorizing the matrix $\mathbf{M}_i^*$ for $\mathbf{G}_i$. Then $\mathbf{M}_2^* = Q\mathbf{M}_1^*Q^T$.

Proof. Construct the respective matrices M_1 and M_2 for $\mathbf{G}_1$ and $\mathbf{G}_2$ accordingly to the proof of Theorem 3. We show that $M_2 = QM_1Q^T$. Indeed:

Denote $\mathbf{P}_i$ as the transition probability matrix of the Markov Chain defined by HYPERFEAT for $\mathbf{G}_i$. As the probability is defined by the distance, which is invariant up to node permutation, we have $\mathbf{P}_2 = Q\mathbf{P}_1Q^T$ and equivalently, $Q^T\mathbf{P}_2Q = \mathbf{P}_1$. In other words, $(\mathbf{P}_2)_{\sigma(w),\sigma(c)} = (\mathbf{P}_1)_{w,c}$.

According to Lemma 9, the Markov Chain defined by HYPERFEAT converges to unique stationary distribution $\pi^{(i)}$. We have: $\pi^{(1)} = \pi^{(1)}\mathbf{P}_1$ and $\pi^{(2)} = \pi^{(2)}\mathbf{P}_2$. As a result $\pi^{(1)} = \pi^{(1)}Q^T\mathbf{P}_2Q$, which leads to $\pi^{(1)}Q^T = \pi^{(1)}Q^T\mathbf{P}_2QQ^T = (\pi^{(1)}Q^T)\mathbf{P}_2$. Therefore, $(\pi^{(1)}Q^T)$ is the unique solution to $\pi = \pi\mathbf{P}_2$. In other words, $\pi^{(2)} = \pi^{(1)}Q^T$. Thus, $\pi^{(2)}_{\sigma(w)} = \pi^{(1)}_w$.

Therefore, according to the proof of Theorem 3, $M_2[\sigma(w)][\sigma(c)] = M_1[w][c]$. As M_i converges to $\mathbf{M}_i^*$, we have $\mathbf{M}_2^*[\sigma(w)][\sigma(c)] = \mathbf{M}_1^*[w][c]$, i.e., $\mathbf{M}_2^* = Q\mathbf{M}_1^*Q^T$.

□

Next, we show that if a hypergraph is only different from the other by a node permutation, HYPERFEAT* obtains the same set of node features for the two hypergraphs.

Lemma 15 (INVARIANCE OF HYPERFEAT*)**.** Suppose that hypergraph $\mathbf{G}_2$ is obtained by applying a node permutation σ on hypergraph $\mathbf{G}_1$. Let $\mathbf{X}_1$ and $\mathbf{X}_2$ denote the matrices of node features extracted by HYPERFEAT* for $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively. $\mathbf{X}_2$ is the result of applying σ on the rows of $\mathbf{X}_1$.

Proof. Denote Q as the permutation matrix corresponding to the permutation σ.

HYPERFEAT* extracts the node feature matrix $\mathbf{X}_i$ for $\mathbf{G}_i$ by factorizing a constant matrix $\mathbf{M}_i^*$ based on Singular Value Decomposition.

According to Lemma 14, $\mathbf{M}_2^* = Q\mathbf{M}_1^*Q^T$.

Let the SVD of $\mathbf{M}_1^*$ be: $\mathbf{M}_1^* = U\Sigma V^T$, then $\mathbf{X}_1$ (by HYPERFEAT*) is equal to $\mathbf{X}_1 = U\sqrt{\Sigma}$. As $\mathbf{M}_2^* = Q\mathbf{M}_1^*Q^T$, the SVD of $\mathbf{M}_2^*$ is: $\mathbf{M}_2^* = QU\Sigma V^TQ^T = (QU)\Sigma(QV)^T$. As a result, by HYPERFEAT*, $\mathbf{X}_2 = (QU)\sqrt{\Sigma} = Q(U\sqrt{\Sigma}) = Q\mathbf{X}_1$.

In other words, $\mathbf{X}_2$ is obtained by applying the permutation σ on the rows of $\mathbf{X}_1$.

□

As a corollary of Lemma 15, HYPERFEAT* can provide sufficient conditions to conclude that two hypergraphs are non-isomorphic. Two hypergraphs are isomorphic if one hypergraph is obtained by applying a node permutation on the other. Hypergraph isomorphism [9] determines if two given hypergraphs are isomorphic, which is a special case of the hypergraph alignment problem where the two hypergraphs are only different by the node permutation. The details are in Theorem 4.

Theorem 4 (HYPERFEAT* AS ISOMORPHISM TEST)**.** If HYPERFEAT* maps the nodes of hypergraphs $\mathbf{H}_1$ and $\mathbf{H}_2$ to two different sets of node embeddings, $\mathbf{H}_1$ and $\mathbf{H}_2$ are not isomorphic.

Proof. Assume that $\mathbf{G}_1$ and $\mathbf{G}_2$ are isomorphic. As a result, there exists a node permutation σ by which $\mathbf{G}_2$ is obtained by applying σ on $\mathbf{G}_1$.

According to Lemma 15, HYPERFEAT* extracts the feature matrix $\mathbf{X}_1$ for $\mathbf{G}_1$ and the feature matrix $\mathbf{X}_2$ for $\mathbf{G}_2$ in which $\mathbf{X}_2$ is obtained by applying σ on $\mathbf{X}_1$ (a reordering of rows). In other words, the sets of node embeddings, by HYPERFEAT*, for $\mathbf{G}_1$ and $\mathbf{G}_2$ are the same.

Therefore, if HYPERFEAT* produces two different sets of node embeddings for $\mathbf{G}_1$ and $\mathbf{G}_2$, we can conclude that $\mathbf{G}_1$ and $\mathbf{G}_2$ are not isomorphic. □

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," 2014.

[2] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[3] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[4] J. S. Vitter, "An efficient algorithm for sequential random sampling," *ACM transactions on mathematical software (TOMS)*, vol. 13, no. 1, pp. 58–67, 1987.

[5] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *WSDM*, 2018.

[6] A. Shiryaev and A. Lysaoff, "Problems in probability," *Springer New York*, 2012.

[7] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," *NeurIPS*, 2014.

[8] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *CIKM*, 2018.

[9] L. Babai and E. M. Luks, "Canonical labeling of graphs," in *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, 1983, pp. 171–183.