Multi-Modal Graph Inductive Learning with CLIP Embeddings (Team 18, Project 78)

Adi Srikanth
NYU Center for Data Science
aks9136@nyu.edu

Andre Chen
NYU Center for Data Science
alc9635@nyu.edu

David Roth
NYU Center for Data Science
dsr331@nyu.edu

Tanya Naheta
NYU Center for Data Science
tsn6109@nyu.edu

Abstract

Multimodal graph-based learning approaches can facilitate a better search experience at Zillow, whose data consists of listing images, descriptions, and other metadata. We aim to use GraphSAGE, an inductive graph representation learning framework, to learn representations using CLIP-initialized node embeddings[3, 10]. Specifically, we train a 2-layer GraphSAGE model using restricted fanout, mean aggregation, ReLU nonlinearity and batch normalization and experiment with three approaches for connecting new, previously unseen nodes to the training graph during inference. Finally, we evaluate updated node embeddings on a cosine similarity-based image-keyword link prediction task and compare their performance to link prediction using embeddings initialized from a fine-tuned CLIP-ViT/32 as a baseline. We find that across several different evaluation datasets at Zillow, GraphSAGE does not conclusively outperform the baseline on link prediction. However, we find that increasing connections to keyword labels improves GraphSAGE performance relative to baseline, and note that on MS-COCO[6], a well-known research dataset with human-generated keyword annotations, GraphSAGE generally outperforms our baseline. Our code is available on Github.

1 Introduction

As web-scale product catalogs grow in size and dimension, the need for effective search and retrieval assumes a more prominent role as a top priority, and often, a primary challenge. This problem is made all the more challenging on datasets that support multiple modalities (e.g. images and text). Zillow, a leading marketplace for real estate and rentals, maintains such a dataset; their database contains a collection of text descriptions and images for the roughly 135 million properties hosted on their platform. Associating images with text descriptions can facilitate search, filtering and indexing over image data and can be critical for surfacing relevant listings to users. In this work, we investigate methods for using aligned image and text encoders (e.g. CLIP) in combination with Graph Neural Networks (GNNs) to extract text attributes from images by framing the problem as a link prediction task over a bipartite graph of images and text attributes.

In particular, we evaluate the precision and recall performance of an inductive GraphSAGE model using CLIP embeddings as node features at attribute detection on the Common Objects in Context dataset [6], as well as a dataset of roughly 80,000 images and 1,500 weakly labeled text attributes made available by the Applied Science and Machine Learning group at Zillow. We find that naive neighborhood mean aggregation and graph convolution approach match but do not conclusively outperform pair-only link prediction performance using Zillow fine-tuned CLIP embeddings. We explore the effects of neighborhood sampling method, model size and embedding dimension on link prediction performance of our model and propose directions for improving on a non-aggregated baseline. Additionally, we develop and evaluate different methods of incorporating unseen nodes into an existing graph to perform inference over new nodes with a trained model.

2 Related Work

A promising, non-graphical approach to multimodal learning currently employed at Zillow is CLIP (Contrastive Language-Image Pre-Training) [10], which uses contrastive learning to generate aligned image and text embeddings for co-occurring image-text pairs. The encoders trained on 400M image-caption pairs demonstrated zero-shot capabilities in image-text matching; We use CLIP encodings to initialize image, and text (keyword and scene) representations for nodes in our graph. We evaluate whether contrastively trained representations from large, open pretraining can benefit from aggregated neighborhood information using graph proximity.

A substantial body of work exists on scalable graph representation learning methods [1, 2, 12, 5, 9, 11]. Deep learning based methods such as Structural Deep Network Embedding [**?**] use multiple layers to capture non-linear network structure and auto-encoders to embed graph nodes. Additionally, Large-scale Information Network Embedding (LINE) extends on graph factorization methods to preserve first and second-order proximities, thereby learning neighborhood network structures [11]. However, most of these methods are inherently transductive, rather than inductive. In a comprehensive survey on graph-based representation learning approaches, Khoshraftar and An describe transductive graph learning techniques as those in which predictions must be made on nodes observed during training, so they do not naturally generalize to unseen data [4]. In contrast, inductive approaches are especially useful when operating on evolving graphs as they generate embeddings on unseen nodes and entirely new subgraphs more efficiently. GraphSAGE is one such inductive approach, making it advantageous for Zillow's core need to constantly update listings and add new information to its platform. Indeed, by learning functions to transform embeddings of new nodes rather than transforming embeddings directly, trained GraphSAGE models can be easily run on new nodes to update their embeddings at inference time [8].

Multimodal graph-based representation learning is well-motivated by applications in search- such as multimodal search queries, thumbnail image selection, and personalization of query results. Misraa et al. use inductive graph representation learning to build a multi-modal retrieval system for Adobe Stock [8], which our project draws key approaches from and expands upon. Specifically, they leverage GraphSAGE in their work to learn aggregation functions over multimodal graph data drawn from the MS-COCO image-keyword annotation dataset that can generalize to new nodes, and introduce a simple method for incorporating new nodes into the original training graph at inference time. The latter component of their work is critical, as new nodes require inbound edges at inference time for GraphSAGE to apply functions learned during training to update their feature representations. In our work, we use a similar framework for training over multimodal data and experiment with three key novel contributions: (1) we initialize image and text node embeddings with CLIP, (2) we evaluate several additional heuristics for connecting new nodes at inference time to the original training graph, and (3) we develop a simple link prediction approach for evaluation of output node representations.

3 Problem Definition and Algorithms

3.1 Task

We aim to take an inductive approach to graph learning, using the GraphSAGE graph convolutional network (GCN) architecture as proposed in Hamilton et al. [3] to learn multimodal representations using graph structure to improve on representations learned by Zillow's CLIP encoder alone.

The inputs to our model are Zillow listing images and two types of textual attributes associated with each image: (1) scenes, which provide a single category describing the overall setting and context of an image (e.g. kitchen), and (2) keywords, which describe objects in the image (e.g. table, granite countertops, etc.). The output of our method are node representations on a multimodal graph of images, scenes, with known linkages between images and associated scenes and/or keywords represented as bidirectional edges. Furthermore, the output graph should have the following properties: (1) the node embedding space should be such that closer nodes in the graph score higher in some similarity metric (e.g. cosine similarity), while distant / unconnected nodes should score lower, and (2) new nodes can be easily added and incorporated into the graph embedding space. To enable quantitative evaluation of our approach, we frame our evaluation as a link prediction task over a bipartite graph of images and known text attributes. We evaluate our model's ability to output high probabilities for true, positive edges and low probabilities for non-existent, negative edges, and compare this ability to Zillow's baseline multimodal approach that does not use graph-based learning.

3.2 Algorithms

3.2.1 GraphSAGE

Given an input graph with nodes, node feature embeddings, and edges, GraphSAGE utilizes a convolutional graph neural network that learns an aggregation function in an unsupervised fashion to infer representations of connected nodes and generalize to nodes not seen during training. In each layer of GraphSAGE, each node samples a number of neighboring nodes, aggregates their feature embeddings via a learnable function, then applies a nonlinear transformation with normalization to produce an updated node representation (Figure 1). Inclusion of additional layers allows a node to receive aggregated features from more distant neighbors in the graph. During training, GraphSAGE uses backpropagation to optimize a contrastive learning objective which seeks to maximize the cosine similarity between nodes which co-occur on a fixed-length random walk and minimize cosine distance between randomly sampled negative edges.

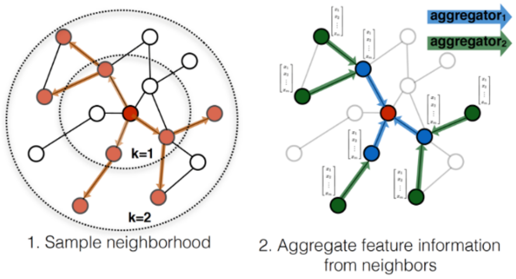


Figure 1: Illustration of GraphSAGE sampling and aggregation process for representation learning

In our implementation, output representations are produced by a two-layer GraphSAGE network with a mean-pooling aggregator function using a fixed number of neighbor nodes, controlled by the "fanout" parameter, at each k-hop distance from the target node. Choices of aggregator function, number of layers, and fanout are particularly critical; we discuss the implications of these decisions in more detail in the experimental evaluation section.

3.2.2 Link Prediction

We measure performance on a link prediction task for node representations before GraphSAGE updates (pre-trained CLIP embeddings) and after GraphSAGE updates to compare performance of GraphSAGE against our baseline.

For efficient computation over all image and keyword nodes in our graph, we compute an NxM cosine similarity matrix between our N image embeddings and M keyword embeddings. We frame the prediction problem as a binary classification task, where for each keyword we predict whether it belongs (1) or not (0) in each of the N images in the validation set based on some cosine similarity threshold. We compute micro and macro-averaged precision, recall, and f1 scores over a range of cosine similarity thresholds, and determine an optimal cosine similarity threshold for CLIP and GraphSAGE based on macro-averaged f1-scores across all keywords.

4 Experimental Evaluation

4.1 Data

Our experimental design requires multimodal datasets that include images, scenes, and keywords. This setup enables the construction of a bi-directional multi-modal graph in which images are connected to both scene and keyword labels. To this end, we use the MS-COCO dataset and a dataset provided by Zillow, both of which we describe in detail below.

4.1.1 MS-COCO Dataset

To complete initial development and validate our experimental pipeline on a more established, human-verified dataset, we use the 2017 MS-COCO image-tag dataset containing 118K images and 80 unique keyword labels [6]. Each image is labeled with a number of keywords, while scene categories for images are obtained using Zillow's proprietary scene labeling algorithm.

4.1.2 Zillow Dataset

After developing our evaluation pipeline using the MS-COCO dataset, experiments are replicated on Zillow data, which contain 83,000 listing images, 18 unique scenes, and 1500 unique keyword attributes. Scene and keywords are attributed to each image using an existing proprietary algorithm at Zillow. Note that relative to the MS-COCO dataset, keyword labels are relatively imbalanced; roughly 20% of the keywords account for 85% of all image labels in our Zillow dataset (in comparison, 65% of keywords in the MS-COCO dataset account for 85% of all image labels). We also retain a smaller Zillow dataset with human-verified scene and keyword labels for final evaluation. This dataset contains 9000 listing images, 30 unique scenes, and 21 unique keywords. With respect to keywords in this dataset, we use only the 10 which co-occur with our training / validation dataset to restrict final evaluation to in-vocabulary keywords. Moving forward, we refer to the smaller Zillow dataset as the "test" set, and the larger Zillow dataset as the "development" set.

4.2 Methodology

4.2.1 Multimodal Graph Construction

We construct a homogeneous graph using the DeepGraph Python library (DGL) [13] with unique images, keywords, and scenes represented as nodes and existing links between images and their respective scenes and keywords as bidirectional edges. We do this for both the test and development datasets independently.

To construct node representations as input to GraphSAGE, we initialize node embeddings using CLIP. For the MS-COCO dataset, we use CLIP's pre-trained image and text encoder with default settings, while for Zillow data we obtain CLIP embeddings from a fine-tuned version of CLIP provided by Zillow. We then partition the development graph into training and validation subgraphs by separating image nodes in the graph into 70% training and 30% validation sets (scene and keyword nodes were kept in both subgraphs). We split in this manner for two reasons: (1) we assume a fixed vocabulary of keywords at Zillow and therefore evaluate using link prediction between new images and in-vocabulary keywords, and (2) Zillow is able to obtain relatively high-fidelity scene labels for new listing images, so we assume scene information is available at inference time.

4.2.2 GraphSAGE Training

To train our GraphSAGE model, we use edge-wise sampling from our training subgraph with a negative sampling ratio of 1:1 to generate batches for input to the model. While we tried several random negative sampling approaches, we found that uniform negative edge sampling (given source node u, choose a node v from the set of all nodes V in the subgraph with equal probability to form a negative edge with) to be the simplest and most effective approach. With respect to model parameters, we tune the model with respect to negative sampling ratio, number of layers, and neighborhood fanout, as we determined these were most impactful to model performance. Table 1 shows hyperparameter tuning results for several top-performing configurations on the Zillow development set. For each configuration, classification metrics are reported at the prediction threshold that produced the highest f1-score.

We select optimal hyperparameters based on macro-averaged precision and therefore use 2 layers, a 1:1 sampling ratio, and fanout of 3 across all GNN layers for evaluation on our Zillow test set. We find that model performance benefits are negligible with > 2 GraphSAGE layers and excessive fanout degrades performance. We considered that a weighted sum of neighbor representations could counteract this effect, but leave exploration of weighting schemes (Graph Attention, Personalized PageRank) for future work.

model	metric	score	n_layers	neg_sampling_ratio	fanout
SAGE	prec@max_recall	**0.000487**	2	1	[3]
SAGE	prec@max_precision	0.001069	3	1	[3]
SAGE	recall@max_recall	0.225256	2	1	[20]
SAGE	recall@max_precision	0.214342	3	1	[5]

Table 1: Hyperparameter settings for the top performing SAGE models. Bold indicates that this was the best performing model under that metric across both SAGE and CLIP.

4.2.3 GraphSAGE Evaluation

A potential use case for multimodal representation learning at Zillow is attribution of keywords from a fixed vocabulary to new listing images in their database. Thus, we evaluate our GraphSAGE-updated node embeddings against baseline CLIP embeddings on image-keyword link prediction.

One major question in our evaluation is whether inclusion of context from the training subgraph improves link prediction performance over our validation subgraph. Thus, at inference time, we experiment with three different approaches of seeding connections between the validation subgraph and training subgraph to provide neighbors to validation nodes for embedding updates. These approaches are outlined below and visualized in Figure 2:

1. **Cosine similarity**: follows a similar approach to one outlined in [8] to connect image nodes from the validation subgraph to the top 5 most similar image and keyword nodes in the training subgraph by cosine similarity
2. **Scene**: connect image nodes from the validation subgraph to the same scene nodes in the training subgraph that they are connected to in the validation subgraph
3. **Self (baseline)**: don't connect validation image nodes to the training subgraph at all, and instead add self-loops for validation image nodes

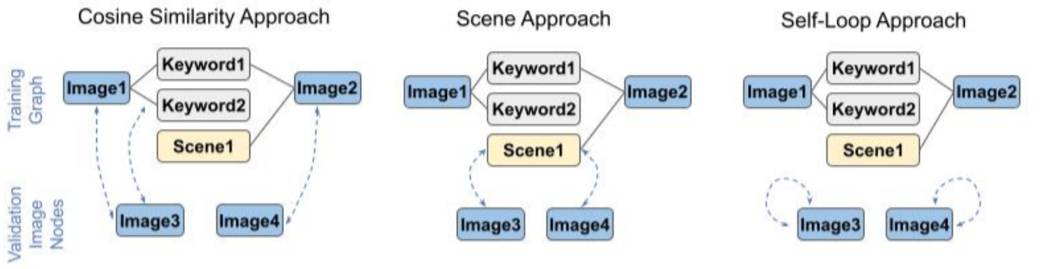


Figure 2: Illustration of different reconnection approaches tested at inference time

We try all three of these approaches with the validation subgraph from the development set as well as the entire graph from the test set and report results below. In all three approaches, we simulate having no knowledge of validation image-keyword links, as would be the case at inference time in a production setting. We hypothesized that the additional context provided in approaches 1 or 2 would yield performance gains in link prediction over approach 3.

4.3 Results

Below we show performance results on our link prediction tasks across our three evaluation datasets (MS-COCO, Zillow development, and Zillow test). For each dataset, we present results for the same four sets of experiments: CLIP (baseline) and graph-based methods using the three node reconnection methods listed above.

Tables 2 and 3 show macro and micro-averaged precision, recall, and f1 scores, with top results for each dataset in bold. Recall that metrics are averaged over 80 keyword classes for MS-COCO, 1500 keyword classes for Zillow development set, and 10 keyword classes for Zillow test set. For each approach, we selected final cosine similarity thresholds for positive prediction based on macro-averaged f1-score. We observe on the Zillow test set that our graph-based approach with cosine reconnection slightly outperforms the CLIP baseline in recall and f1. while other approaches are similar in performance to, but fail to outperform, the baseline. On MS-COCO, graph-based approaches achieve better recall while the baseline achieves better precision at our optimal thresholds. On the Zillow development set, it appears that our graph approaches significantly underperform our baseline, which we investigate in our discussion section.

Table 2: Macro-averaged link prediction classification metrics

Dataset	**Method**	**Precision (Macro)**	**Recall (Macro)**	**F1 (Macro)**
Zillow Development	sage_cosine	0.000457	0.193178	0.000913
Zillow Development	sage_scene	0.000419	0.046864	0.00083
Zillow Development	sage_self	0.006819	0.222706	0.013234
Zillow Development	clip	**0.008516**	**0.399308**	**0.016677**
Zillow Test	sage_cosine	0.347497	**0.715822**	**0.467868**
Zillow Test	sage_scene	0.361937	0.587978	0.448063
Zillow Test	sage_self	0.358647	0.592516	0.44683
Zillow Test	clip	**0.36339**	0.605616	0.454228
MS-COCO	sage_self	0.33162	0.560273	0.416637
MS-COCO	sage_scene	0.344293	0.535681	0.419175
MS-COCO	sage_cosine	0.326984	**0.632623**	0.43113
MS-COCO	clip	**0.547829**	0.457524	**0.498621**

Table 3: Micro-averaged link prediction classification metrics

Dataset	**Method**	**Precision (Micro)**	**Recall (Micro)**	**F1 (Micro)**
Zillow Development	sage_cosine	0.000004	0.000174	0.000008
Zillow Development	sage_scene	0.000005	0.000051	0.000008
Zillow Development	sage_self	**0.000069**	0.000469	0.00012
Zillow Development	clip	0.000066	**0.000492**	**0.000117**
Zillow Test	sage_cosine	0.035999	**0.061335**	0.045369
Zillow Test	sage_scene	0.038723	0.054413	0.045246
Zillow Test	sage_self	0.038353	0.054545	0.045038
Zillow Test	clip	**0.039976**	0.056326	**0.046763**
MS-COCO	sage_self	0.005677	0.005547	0.005611
MS-COCO	sage_scene	0.005981	0.005074	0.00549
MS-COCO	sage_cosine	0.005566	**0.006929**	**0.006173**
MS-COCO	clip	**0.007388**	0.003791	0.005011

Figure 3 provides ROC curves to demonstrate link prediction performance over a range of possible cosine similarity thresholds. We observe that on the MS-COCO dataset, our "self" and "scene" based reconnection methods slightly outperform CLIP, while on the Zillow test dataset, all three graph-based approaches appear to be on par with CLIP. On the Zillow development set, reconnection methods that incorporate neighborhood information from the training graph perform worse than random guessing, while the self-loop approach nearly performs as well as the baseline method.

Figure 4 (Appendix) shows a comparison of precision and recall by keyword for our best graph-based approach (green) against our baseline CLIP method (pink) on top 10 most frequent keywords at each model's optimal prediction threshold. The keywords shown are sorted from top to bottom in order of decreasing prevalence. We observe that performance varies significantly across keywords. Note that we only show results by keyword for the top-performing reconnection approach for our graph-based method (cosine for Zillow test data, scene for MS-COCO data).

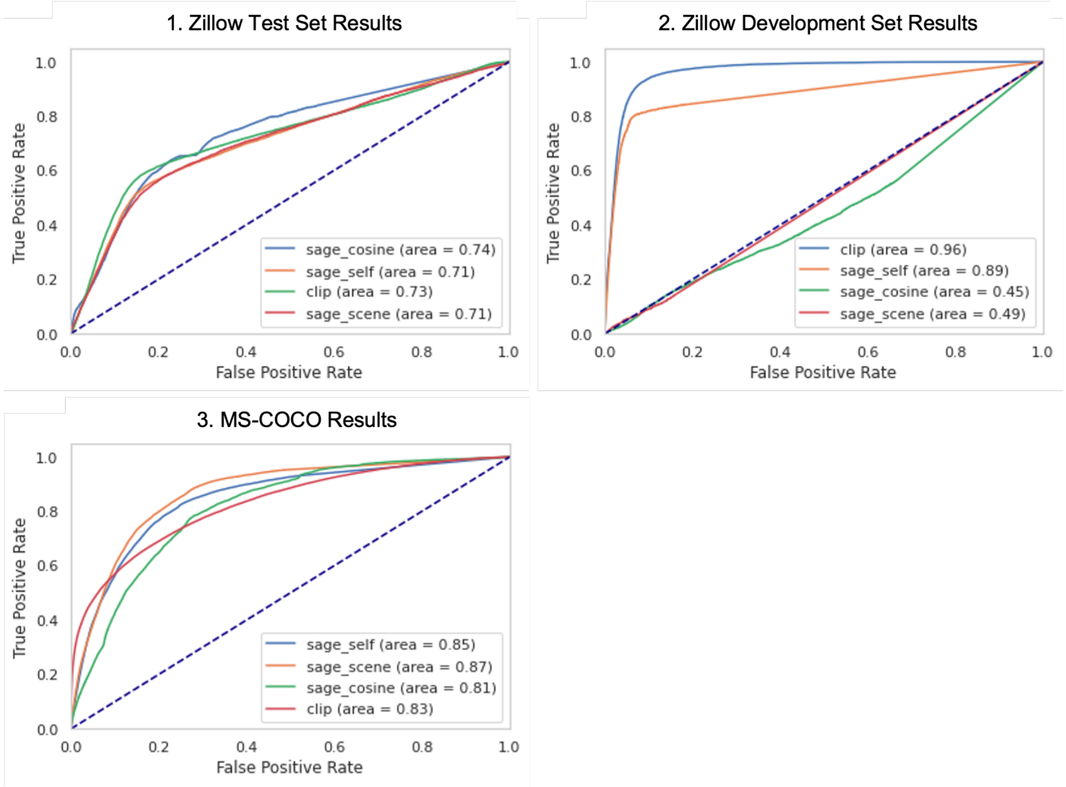


Figure 3: Link Prediction ROC Curves for Zillow Test Set (1), Zillow Development Set (2), and MS-COCO (3)

4.4 Discussion

We recall the two key hypotheses outlined earlier in our report: (1) learned GraphSAGE embeddings would outperform CLIP embeddings as measured by link prediction performance, and (2) GraphSAGE embeddings would improve on link prediction using only self-loops as seeded validation edges.

To summarize, we are not able to definitively confirm hypothesis 1 or hypothesis 2 in totality based on our experimental results. However, we explore some promising observations in this section with respect to both. Additionally, we offer concrete next steps in our conclusion for further exploration that may yield clearer results.

First we discuss the role of noisy training labels by comparing results across datasets. In evaluation link prediction results for the MS-COCO dataset, we note that graph-based approaches generally yield better recall (micro and macro-averaged) and f1 (micro-averaged) scores across all three methods of node reconnection. On Zillow's test set, graph-based approaches still demonstrate strong performance, though results are more comparable to the baseline.

However, the benefit of graph-based approaches becomes less clear upon analyzing performance on Zillow's development dataset. On Zillow's weakly labeled development set, all three versions of our approach underperform CLIP in nearly every metric (and notably in micro and macro F1). Figure 3 also shows that the self-loop method significantly outperforms the other two reconnection methods on the Zillow development set. A possible explanation for performance inconsistencies across datasets is the varying fidelity of keyword and scene labels across datasets. Notably, MS-COCO and the Zillow test set have human-generated keyword labels, while the Zillow development set uses significantly noisier machine-generated keyword labels.

This agrees with our theoretical understanding of our GraphSAGE implementation. Because node embedding updates depend on neighboring nodes at various depths during GraphSAGE forward inference, and keyword nodes generally have high in-degrees in our graphs, we expect noisier keyword nodes to degrade performance. In short, CLIP appears to be more robust to noisy data while the GraphSAGE approach may exhibit a higher ceiling for performance, albeit contingent upon a reliably labeled dataset. Additional experiments introducing noise into MS-COCO keyword labels may also help verify this finding, but we leave these to future research. We also speculate on the effect of scale; In the original CLIP paper, robustness to noisy annotations is attributed to the scale of the training dataset (4e7 annotated images), while our training datasets are on the order of 1e5 annotated images.

To understand differences between different reconnection methods in our graph-based approach, we visualize node embeddings before and after GraphSAGE updates by projecting a sample of positive and negative node pairs from the Zillow test set into 2D space using UMAP[7]. Additionally, for the positive and negative pair samples, we plot changes in cosine similarity between inter-pair node embeddings. Results are found in Figure 5 (Appendix). We can see through our UMAP plots and associated bar plots that after GraphSAGE updates, updated embeddings of connected nodes are generally more similar than those of unconnected nodes, which aligns with our expectations given GraphSAGE's contrastive learning objective. However, we do not observe significant differences in the UMAP plots or associated bar plots across the three reconnection approaches, which may explain why we do not see pronounced differences between our reconnection methods in link prediction performance.

Ultimately, while we do not find our approach ready for deployment in its current state, we offer specific next steps in the following section to explore potential design improvements.

5 Conclusions

The most significant takeaway from our work is the idea that not only can multiple modes of data cooperate within a single store, but also that graph based learning can fuel more precise image and text embeddings moving forward. Moreover, we see that our contrastive learning loss function successfully results in learned embeddings that move connected nodes closer and unconnected (or distantly connected) nodes further in embedding space. We hope further data enrichment (especially connections between nodes) will yield improved performance- a result that informs this project and others as well.

Of course, we caveat the takeaways of our work in multiple ways. Most importantly, while we have reason to believe that optimized graph based learning can yield useful results, our experiment here does not give conclusive evidence to the effect. Given the inconsistencies in our results across datasets, additional experimentation is required as a prerequisite to solid conclusion. Additionally, our approach to our GNN can still be further optimized. These shortcomings, however, can certainly be rectified. Next, we provide a roadmap for addressing these caveats en route to pursuing a more definitive final result.

As a first step, we recommend employing strong regularization in updating node embeddings. This would help address some of the sparsity that resulted from implementing contrastive loss and would also make our solution more robust to noisy data and more consistent across datasets. Regularization can come in simple forms such as GNN dropout, but can also be implemented more granularly with a method such as enforcing a limit on "sparse" entries (entries less than some threshold) allowed in each embedding. Additionally, we recommend the utilization of edge weights in the GraphSAGE implementation. These weights can present node connections probabilistically instead of forcing a dichotomous representation. Consequently, this can help mitigate impacts of noisy data by capturing uncertainty. Finally, we also recommend importing additional node and edge types to capture more types of relationships within Zillow's data. For example, connecting nodes by attributes such as home price or date uploaded would offer additional forms of neighborhood data for GraphSAGE to learn from and use in link prediction.

In addition to thoroughly addressing the aforementioned action items, there is remaining room to explore within this general topic. Additional future work could include implementing multi-modal search (where a user can search for a listing using images and/or text) or fine-tuning recommender systems to suggest listings to users. Otherwise, there are many practical applications to choose from as a follow-up to this effort.

6 Lessons Learned

Here, we focus on two lessons that were instrumental for our team during this effort. One, our team was able to identify a need for technical standardization and address it through version control and shared coding principles. Two, our team was able to rapidly iterate in order to receive concrete feedback and progress. Below, we elaborate slightly on both lessons.

As we began developing solutions for this project, we noticed discrepancies in how our code was organized and written. This included directory management, file formats, and maintenance of our GitHub repository. As our codebase began to approach a level of unwieldiness, we met as a team and established a consistent environment (including package versions, directory structure) across our team. We also established norms for our repository and modularized our code into accessible scripts with documented command-line arguments. We felt that these were effective solutions and will be carried into future projects.

At the onset of our project, we were primarily equipped with our end goal (in the form of our project statement). As such, we developed work with that goal in mind. However, this resulted in large chunks of work and did not allow for frequent feedback from our mentors at Zillow. To adapt, we began to set smaller goals at a weekly cadence and incorporated feedback from our weekly meetings with the team at Zillow. Again, this was an effective change and would be welcome in similar settings in the future.

7 Student Contributions

David Roth: Contributed code to GraphSAGE implementation/training, DGL graph construction, hyperparameter evaluation and model deployment on NYU's cluster. Also contributed to report writing and typesetting.

Andre Chen: Contributed significant code to generate MS-COCO CLIP embeddings, build MMKGs for MS-COCO and Zillow datasets, implement GraphSAGE model training, compute metrics, and plot results. Also contributed to report writing and poster presentation.

Tanya Naheta: Researched related work and provided initial design recommendations and baseline metrics. Also contributed to report writing and final poster presentation.

Adi Srikanth: Contributed code to GraphSAGE implementation, basic metrics generation, scene-reconnection experiment, and link prediction validation method. Also contributed to report writing and poster presentation.

8 References

References

[1] Shaosheng Cao, Wei Lu, and Qiongkai Xu. GraRep: Learning graph representations with global structural information, 2015. URL `https://doi.org/10.1145/2806416.2806512`.

[2] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks, 2016. URL `https://arxiv.org/abs/1607.00653`.

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017. URL `https://arxiv.org/abs/1706.02216`.

[4] Shima Khoshraftar and Aijun An. A survey on graph representation learning methods, 2022. URL `https://arxiv.org/abs/2204.01855`.

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016. URL `https://arxiv.org/abs/1609.02907`.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context, 2014. URL `https://arxiv.org/abs/1405.0312`.

[7] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018. URL `https://arxiv.org/abs/1802.03426`.

[8] Aashish Kumar Misraa, Ajinkya Kale, Pranav Aggarwal, and Ali Aminian. Multi-modal retrieval using graph neural networks, 2020. URL `https://arxiv.org/abs/2010.01666`.

[9] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations, 2014. URL `https://doi.org/10.1145/2623330.2623732`.

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`.

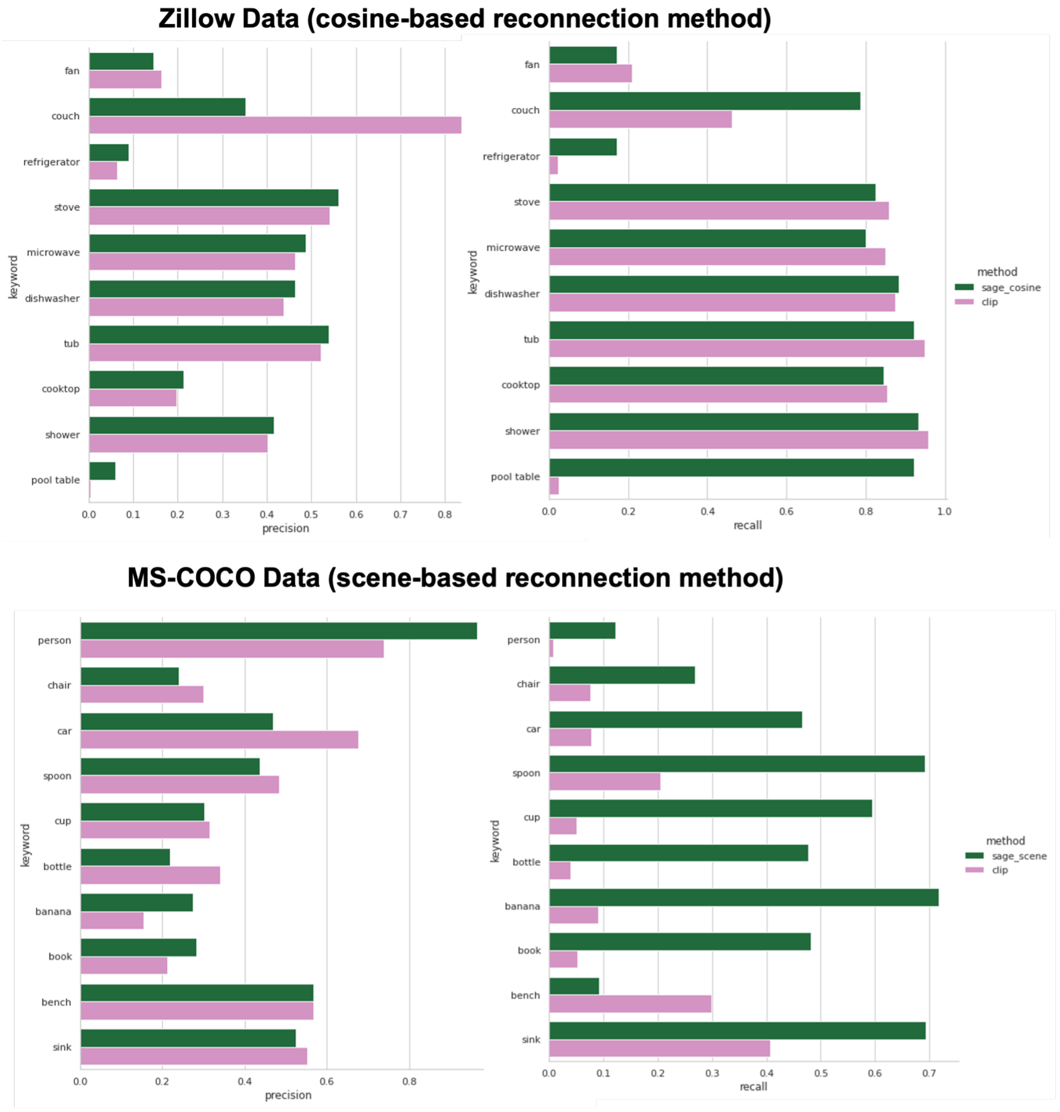
[11] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale information network embedding, 2015. URL `https://doi.org/10.48550/arXiv.1503.03578`.

[12] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding, 2016. URL `https://doi.org/10.1145/2939672.2939753`.

[13] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2019. URL `https://arxiv.org/abs/1909.01315`.

A Appendix

Below we present additional figures from our experiments to supplement findings and key insights from section 4.



Zillow Data (cosine-based reconnection method)

MS-COCO Data (scene-based reconnection method)

Figure 4: Classification metrics (precision and recall) by keyword

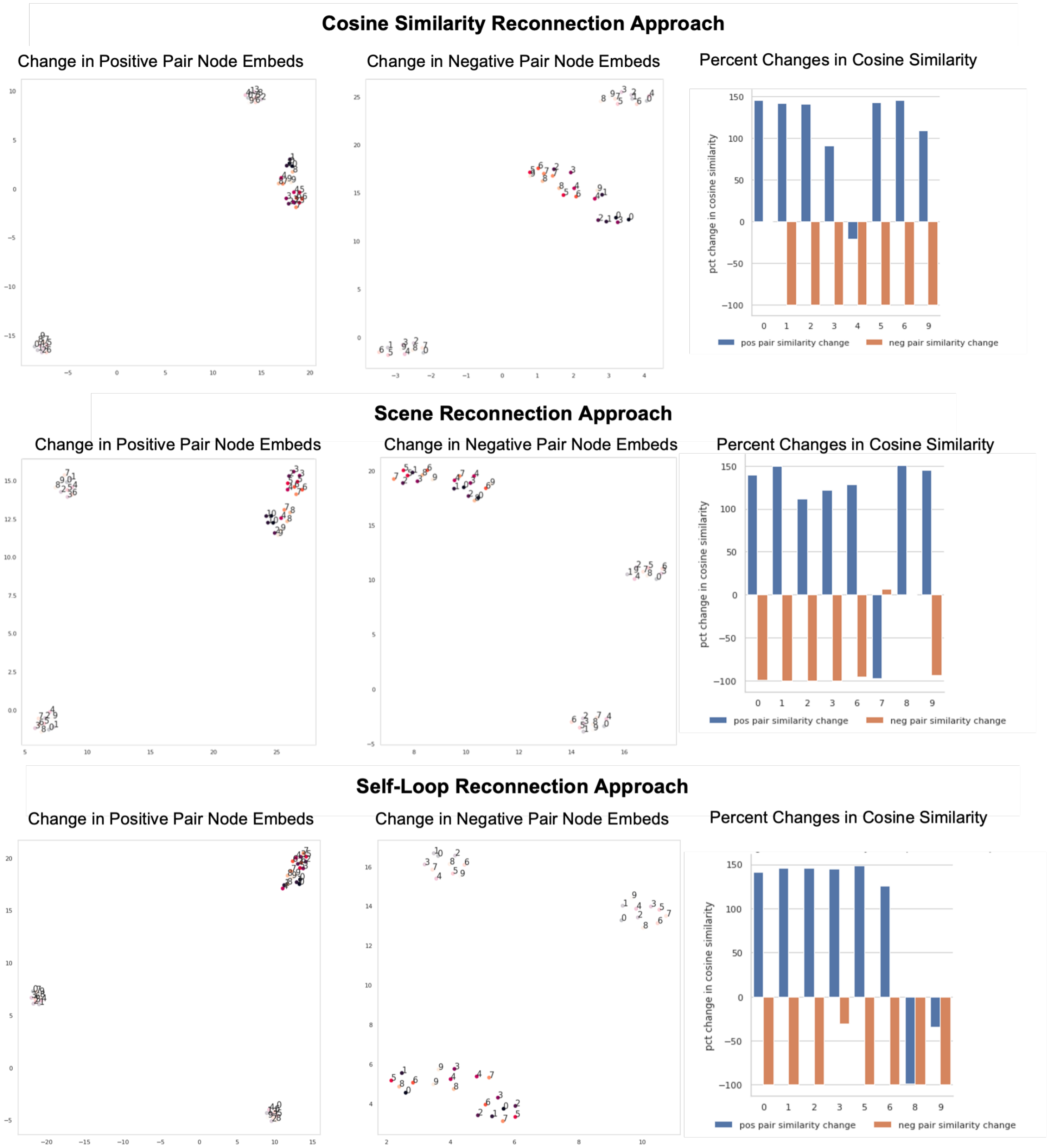


Figure 5: Analysis of 10 randomly sampled positive and negative node pairs from the Zillow test set graph before and after GraphSAGE updates. In each row, the left two plots show 2D UMAP projections of embeddings before (light shade) and after (dark shade) GraphSAGE updates for positive, connected node pairs (first plot) and negative, unconnected node pairs (second plot). The right-most plot in each row shows change in cosine similarity for positive and negative node pairs after applying GraphSAGE updates. In both the first and second pair plots, node pairs have the same color and number label.