Adaptive Graph-Based Word Representations (AGWR)

Jose Cruzado
jcruzado@uchicago.edu

Abstract

In this work, we propose a novel approach for learning contextual word embeddings by leveraging syntactic dependencies and multi-relational graph structures. Our method integrates three distinct types of graphs: a global co-occurrence graph capturing word co-occurrence statistics, a syntactic graph representing sentence-level syntactic dependencies, and a similarity graph where edge weights reflect cosine similarity between pre-trained FastText embeddings. We construct a unified multi-relational graph that encompasses the three types of graphs mentioned. A Relational Graph Attention Network (RGAT) is then employed to perform relation-specific linear transformations and attention-based message passing. A masked language modeling style loss is used during training, encouraging the model to reconstruct the original embeddings for masked words while integrating contextual information from diverse relational signals. Our approach demonstrates competitive performance on the Word-in-Context (WiC) dataset, achieving accuracy comparable to ELMo despite being trained on a significantly smaller corpus. Additionally, we evaluate our embeddings through semantic clustering, showing that they effectively capture contextual distinctions between word senses. Our findings highlight the potential of multi-relational graph structures for contextual embedding learning and more efficient alternatives to transformer-based models. [1]

1 Introduction

Recent advances in natural language processing (NLP) have been largely driven by models that learn rich contextual representations, as evidenced by transformer-based architectures such as BERT [5] and GPT [17]. These models have revolutionized NLP tasks by capturing deep semantic and syntactic properties through self-attention mechanisms. Despite their success, transformers primarily rely on sequential attention mechanisms, making them inherently dependent on large-scale training data and significant computational resources. This limitation raises concerns about efficiency, scalability, and accessibility, especially in resource-constrained environments.

In parallel, graph neural networks (GNNs) have emerged as a powerful tool for modeling non-Euclidean data structures, with methods such as Graph Convolutional Networks (GCNs) [8] and Graph Attention Networks (GATs) [20] demonstrating notable success in diverse domains such as chemistry, social networks, and knowledge graphs. By leveraging structured representations and relational dependencies, GNNs offer an alternative paradigm for capturing word relationships that extend beyond traditional sequence-based models.

Language, however, is inherently structured by complex relationships such as word co-occurrence, syntactic dependencies, and semantic similarities. Leveraging these relationships explicitly can offer complementary insights to those provided by traditional sequential models. Motivated by this

[1] All results presented in this document can be found and reproduced using the code available at: https://github.com/josecruzado21/AGWR.

observation, we propose a novel framework for learning contextual word embeddings by integrating multi-relational graph structures. Our approach constructs three distinct graphs from text:

1. **A global co-occurrence graph** that captures statistical word associations, providing corpus-wide perspective on how words relate based on proximity.
2. **Sentence-level syntactic graphs** that encode grammatical dependencies, offering insights into the hierarchical structure of language
3. **Similarity graphs** where edge weights reflect cosine similarity between pre-trained FastText embeddings [3], enhancing representations with external semantic information

By merging these heterogeneous graphs into a unified multi-relational graph, we aim to create embeddings that reflect the diverse linguistic signals present in natural language. To achieve this, we employ a Relational Graph Attention Network (RGAT)[4], an extension of GNNs designed to handle multi-relational data. The RGAT model allows us to apply relation-specific transformations and attention-based message passing, enabling the model to selectively incorporate information from different relationship types. This mechanism ensures that word representations are context-sensitive, adapting dynamically based on surrounding relational signals.

To train our model, we use a masked reconstruction loss, inspired by the masked language modeling (MLM) objective of BERT. This encourages embeddings to incorporate contextual signals while preserving the intrinsic meaning of words. Unlike traditional static word embeddings, our approach learns representations that evolve based on context, making them more suitable for tasks requiring nuanced language understanding.

Our key contributions are threefold:

1. **A novel multi-relational graph-based framework** that integrates co-occurrence, syntactic, and semantic similarity information into a single unified structure.
2. **A relation-aware GNN architecture** leveraging RGAT to effectively aggregate and differentiate between heterogeneous linguistic signals.
3. **Empirical validation on multiple NLP tasks**, including word sense disambiguation and semantic clustering, where our approach demonstrates improved contextual sensitivity over static embeddings and performs competitively with transformer-based methods.

This study builds upon existing research in graph-based NLP while offering a complementary perspective to transformer-based models. By explicitly modeling the relational structure inherent in language, our approach highlights the potential of GNNs as a compelling alternative for contextual representation learning, particularly in scenarios where computational efficiency is a priority.

2 Related Work

Over the years, significant progress has been made in learning word representations, with early methods relying on purely distributional properties extracted from large corpora. Traditional models such as Word2Vec [12] and GloVe [14] have been widely adopted due to their efficiency in capturing semantic relationships through co-occurrence statistics. Word2Vec, for instance, learns embeddings using local context windows, optimizing for a predictive objective that distinguishes between words appearing in the same neighborhood and randomly sampled negative examples. GloVe, on the other hand, constructs a global word co-occurrence matrix and factorizes it to obtain vector representations that encode corpus-wide statistical patterns. While these models generate effective word representations, they have a fundamental limitation: their embeddings are static, assigning each word a single vector regardless of its varying meanings in different contexts.

The rise of contextual word embeddings marked a significant advancement over static representations. Models like ELMo [15], BERT, and their subsequent transformer-based extensions introduced embeddings that dynamically adjust based on surrounding words, making them particularly powerful in handling polysemy and subtle semantic shifts. ELMo leverages deep bidirectional LSTMs trained with a language modeling objective, generating word vectors that incorporate both left and right context. BERT further advances this paradigm with a masked language model that conditions on bidirectional context, leading to embeddings that encode richer syntactic and semantic information.

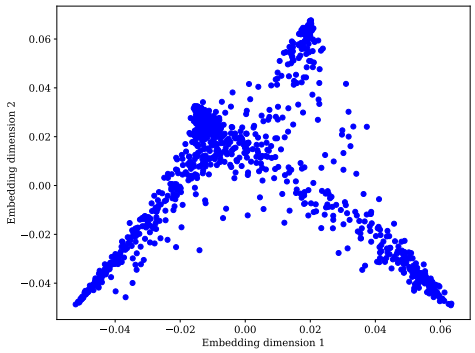
Despite their effectiveness, these models rely on sequential architectures that do not explicitly leverage graph-based relational structures, which could provide additional structural information beyond co-occurrence patterns. Moreover, while self-attention mechanisms capture dependencies between words, they do not inherently model structured relationships such as syntactic dependencies or hierarchical word relations, which could further enhance contextual embeddings.

2.1 Graph-Based Word Embeddings

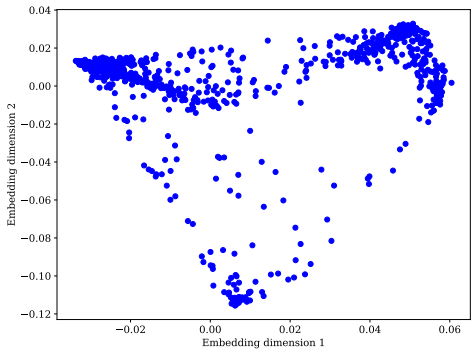
An alternative approach to improving word representations is the use of graph-based models, which explicitly encode word relationships beyond simple co-occurrence windows. Some of the earliest graph-based methods did not rely on deep learning but instead utilized spectral techniques and matrix factorization to construct meaningful word embeddings.

One of the foundational works in this space, and the original inspiration for this project, was introduced by Belkin and Goldsmith [2]. Their work applied spectral decomposition on a nearest-neighbor graph constructed from word contexts. In their approach, words were represented as nodes in a graph, with edges connecting words that appeared in similar left or right contexts. By computing the eigenvectors of the normalized Laplacian, they extracted lower-dimensional embeddings that preserved the structural relationships in the graph. This method provided a novel perspective on distributional semantics but ultimately suffered from the same limitation as early word embedding models—producing static representations that do not adapt to varying contexts.

To further understand the effectiveness of this method, we conducted a replication of Belkin and Goldsmith's approach using modern computational tools. The results obtained closely matched those presented in the original study, reaffirming the efficiency of Laplacian-based embeddings in capturing meaningful word relationships. One key takeaway from our replication was the remarkable computational efficiency of spectral decomposition with today's hardware, we were able to compute embeddings in a matter of seconds. The code for this replication, along with detailed implementation steps, can be found in the GitHub repository linked to this project. Below, we present a visual representation of the left and right embeddings in two dimensions:



(a) Replication of 2D left embeddings. The location of non-finite and finite verbs, nouns, and prepositions aligns with Belkin and Goldsmith's findings.



(b) Replication of 2D right embeddings. The spatial organization of adjectives and prepositions remains consistent with the patterns observed in the original study

Beyond spectral methods, other early graph-based approaches sought to enhance word embeddings by incorporating linguistic structure through probabilistic modeling. A notable example is Probabilistic Embeddings with Laplacian Graph Priors (PELP), which integrates graph-based regularization into static embeddings [21]. By applying Laplacian priors, the model forces words that share certain relationships—such as synonymy or co-definition in lexical resources—to be mapped closer in the embedding space. While this framework enables word embeddings to encode structured relationships, it still produces non-contextual representations, failing to capture dynamic variations in meaning.

A different line of research has explored hierarchical structures in word embeddings, recognizing that words exist in a structured ontology rather than as isolated points in a continuous space. The work of Guo and Skiena introduces directed hierarchical embeddings, where words are ordered based on their

"entity power," linking each word to its closest more powerful neighbor [6]. This approach effectively uncovers hypernym relations and latent taxonomies, but it is primarily concerned with organizing embeddings rather than dynamically adapting them to new contexts. Similarly, Graph-Glove extends the GloVe algorithm by embedding words in a weighted graph, where distances are defined by shortest paths rather than traditional co-occurrence metrics [18]. This allows words to be embedded in a space that more naturally reflects semantic relationships, yet it still lacks context-dependent adaptability.

Recent advancements have turned to Graph Neural Networks (GNNs) to learn richer, more adaptive word representations by leveraging relational structures in an end-to-end manner. Traditional embeddings treat words as independent entities, whereas GNN-based approaches allow message passing between connected words, enabling each word's representation to be updated dynamically based on contextual relationships. Models such as Graph-based Dynamic Word Embeddings (GDWE) continuously update word representations by incorporating information from Word-Level Knowledge Graphs (WKGs) [11].

Other efforts, such as Graph-based Relation Mining (GRM), address the challenge of out-of-vocabulary (OOV) word embeddings by constructing Word Relationship Graphs (WRGs) [10]. These graphs connect OOV words to semantically related words through shared morphemes or structural similarities, offering an efficient alternative to conventional subword decomposition methods.

Beyond capturing semantic relationships, syntactic structure has also been incorporated into graph-based word embeddings. One such model, Graph-based Syntactic Word Embeddings, constructs syntactic graphs from constituency parse trees and applies random walk methods to learn representations that reflect grammatical roles [1]. Unlike dependency-based embeddings, which model head-dependent relations, constituency-based embeddings encode hierarchical phrase structure, allowing them to capture fine-grained syntactic distinctions. This makes them particularly useful for tasks such as POS tagging, question classification, and grammatical disambiguation.

Another approach focuses on leveraging graph structures for contextual embedding learning. The Word-Graph2Vec model, for example, constructs a word co-occurrence graph and applies random walks to generate training sequences for a skip-gram model [9]. By treating word embedding learning as a graph-based sequence prediction task, Word-Graph2Vec achieves higher efficiency than traditional neural embeddings, making it particularly suitable for large-scale corpora.

Our work builds upon these prior approaches by integrating multiple relational graphs into a single unified framework. Unlike existing models that focus on a single graph type (co-occurrence graphs, syntactic graphs, or knowledge graphs), we combine a global co-occurrence graph, a syntactic dependency graph and a semantic similarity graph.

Moreover, we leverage Relational Graph Attention Networks (RGATs) to perform relation-specific transformations and dynamic attention-based message passing, ensuring that our embeddings adapt to context while preserving linguistic structure.

3 The dataset

3.1 Brown Corpus

We utilize the Brown Corpus, an electronic collection of American English text samples, considered the first structured corpus covering a diverse range of genres. The corpus contains approximately 1 million words, spread across 50,000 sentences, with a total vocabulary of around 60,000 unique words. This dataset is publicly available and can be accessed through the NLTK Python package. Below is an example excerpt from the corpus:

```
The Fulton County Grand Jury said Friday an investigation of
Atlanta's recent primary election produced no evidence that
any irregularities took place.
```

This dataset includes Part-of-Speech (POS) tagging, a linguistic annotation that assigns a grammatical category (e.g., noun, verb, adjective) to each word in a sentence. Initially, we considered leveraging these POS tags to construct syntactic graphs, but ultimately decided against it. Automating syntactic dependency inference solely from POS tags would require either:

1. Oversimplifying syntactic relationships (e.g., assuming all adverbs modify the nearest verb)
2. Manually annotating dependencies, a process that is infeasible given the corpus size.

As a result, we did not use POS tagging in our model, opting instead for Stanza's dependency parsing tools [16]. Below is an example of POS-tagged text from the Brown Corpus:

```
The/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl
said/vbd Friday/nr an/at investigation/nn of/in Atlanta's/np$
recent/jj primary/nn election/nn produced/vbd ``/`` no/at
evidence/nn ''/'' that/cs any/dti irregularities/nns took/vbd
place/nn ./.
```

The tag following each word denotes its POS category (e.g., *nn* for noun, *vbd* for past-tense verb). For more details on POS tagging, refer to [7].

3.2 Word-in-Context (WiC) dataset

To evaluate our contextual embeddings, we employ the Word-in-Context (WiC) dataset, a widely used benchmark for word sense disambiguation (WSD). WiC assesses whether a given word carries the same meaning across two distinct sentences. Below is an example:

Sentence 1	**Sentence 2**	**Target Word**	**Label**
The bank of the river was flooded.	He deposited money in the bank.	bank	False
She plays the violin beautifully.	He decided to play soccer.	play	False
The painting was hung on the wall.	The kids painted the wall yesterday.	wall	True

Table 1: Example Entries from the WiC Dataset

Here, Sentence 1 and Sentence 2 contain the target word, whose meaning is evaluated. The Label column indicates whether the word carries the same sense (True) or a different sense (False) in both sentences.

The dataset consists of 5,428 training examples and 638 validation examples.

3.3 Data processing

Before constructing our graphs, we preprocess the corpus by:

1. Filtering the top 25,000 most frequent words to serve as graph nodes.
2. Excluding sentences shorter than 3 words to ensure meaningful relational structures.

3.4 Graph generation

3.4.1 Co-occurrence graph

The co-occurrence graph is an undirected, unweighted graph constructed for the entire dataset. Nodes represent words, and edges exist if two words co-occur within a fixed-size sliding window. This approach differs from the method proposed by Belkin and Goldsmith (2002), which treats left and right contexts separately and considers only immediate co-occurrence.

For example, in our case, given the sentence "I play soccer", a window size of 2 creates the following graph:

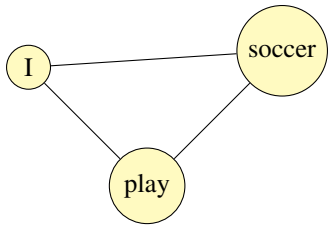


Figure 2: Co-occurrence graph for "I play soccer" with a window size of 2.

3.4.2 Syntactic graphs

Unlike the co-occurrence graph, syntactic graphs are sentence-specific. Initially, we considered deriving syntactic relationships from POS tagging, but automating this proved challenging. Instead, we use Stanza [16], a dependency parser trained on Universal Dependencies (UD) formalisms [13]. The syntactic graphs are also unweighted and undirected.

Below is an example of how a syntactic graph looks for the sentence "I play soccer"

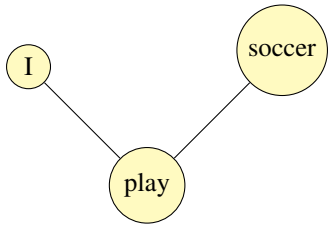


Figure 3: Syntactic graph representation of "I play soccer." Dependencies: "I" (subject) → "play"; "soccer" (object) → "play."

3.4.3 Similarity graphs

These are undirected, weighted graphs where edge weights represent cosine similarity between word embeddings (FastText, 300-dimensional vectors). Each sentence has its own similarity graph. Again, below an example of similarity graph for the sentence "I play soccer"

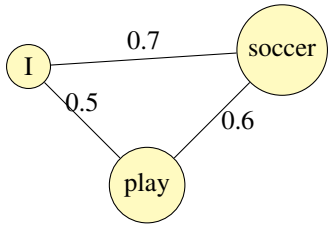


Figure 4: Similarity graph representation of "I play soccer" with cosine similarity weights.

4 Method

4.1 Overview

Our methodology is inspired by the Relational Graph Attention Network (RGAT) proposed by Busbridge et al. [4], which extends the Graph Attention Network (GAT) by incorporating multi-relational graph structures. Unlike standard GAT, which assumes a homogeneous graph structure with a single edge type, RGAT allows:

- **Multiple edge types**, enabling relation-aware message passing.
- **Edge attributes**, allowing attention mechanisms to incorporate additional structural information.

- **Relation-specific transformations**, where different edge types use distinct weight matrices.

The architecture proposed by the authors in the RGAT paper is as follows:

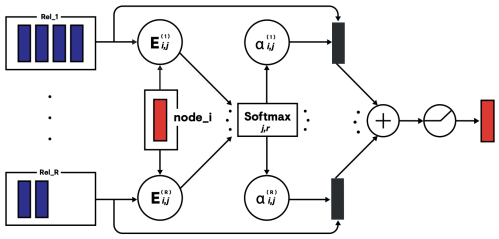


Figure 5: ARGAT architecture as proposed in Relational Graph Attention Networks paper

The figure above illustrates the ARGAT architecture, which extends the concept of graph attention to handle relational graphs by applying attention mechanisms across different relation types. The attention scores are computed independently for each relation type, but the softmax operation is applied globally across all relations and nodes in the neighborhood. This means that the attention coefficients $\alpha_{i,j}^{(r)}$ for node i are normalized across all relations r and neighboring nodes j, allowing the model to dynamically weigh the importance of nodes and relations in a unified manner.

In this work, we define a multi-relational graph as $G = (V, E, R)$, where:

- V is the vocabulary, consisting of $|V|$ words.
- E is the set of edges, representing relations between words.
- R is the set of relation types, including:
  - **Co-occurrence relations**: Capturing statistical associations between words.
  - **Syntactic relations**: Encoding grammatical dependencies.
  - **Semantic similarity relations**: Representing cosine similarity between embeddings.

4.2 Relational Graph Attention Convolutional Layer

Each RGAT layer performs transformations through an attention-based message-passing mechanism. The computation proceeds as follows:

Although our graph is undirected, we maintain separate indexing for inbound and outbound nodes to apply distinct transformations.

4.2.1 Initial Embeddings

We initialize a matrix X of shape $|V| \times 300$, where:

- $|V|$ is the vocabulary size.
- Each row corresponds to the 300-dimensional embedding of a word.

4.2.2 Key and Query Transformations

We start by transforming the inbound (X_i) and outbound (X_j) embedding vectors using a weight matrix $W^{(r)}$ for $r = 1, 2, 3$:

$$V_i^{(r)} = W_1^{(r)} X_i \qquad W_j^{(r)} = W_1^{(r)} X_j \tag{1}$$

Both inbound and outbound nodes, for each type of relationship, are transformed into their respective latent representations using the same matrix $W_1^{(r)}$. Since the graph is undirected, the transformed representations V_i and W_j encode equivalent information; however, their matrices are permuted due to the fact that each node appears as an inbound node as many times as it does as an outbound node.

Then, for each node, we compute query and key vectors:

$$q_i = V_i^{(r)} Q, \qquad k_j = W_j^{(r)} K \tag{2}$$

where:

- $Q \in \mathbb{R}^{(H*d_{out})\times(H*d_k)}$ and $K \in \mathbb{R}^{(H*d_{out})\times(H*d_k)}$ are learnable projection matrices.
- H is the number of attention heads.
- d_{out} is the output embedding dimension.
- d_k is the key-query space dimension.

4.2.3 Edge Attribute Processing

We integrate edge weights into the attention mechanism. In the case of co-occurrence and syntactic edges, the attribute $e_{i,j}^{(r)}$ is a binary indicator (presence or absence of an edge). However, for similarity edges, $e_{i,j}^{(r)}$ is computed using cosine similarity. To incorporate edge attributes into the attention mechanism, we transform them as follows:

$$e_{i,j}^{(r)*} = e_{i,j}^{(r)} W_2 \tag{3}$$

where W_2 is a learnable transformation matrix. We further refine the edge attributes via:

$$e_{i,j}^{(r)**} = e_{i,j}^{(r)*} E \tag{4}$$

where E maps edge attributes into a scalar space for attention computation.

4.2.4 Attention Computation

The attention coefficient for each edge is computed as:

$$a_{i,j}^{(r)} = \text{LeakyReLU}(q_i^{(r)} + k_j^{(r)} + e_i^{(r)**}) \tag{5}$$

The normalized attention scores are obtained via:

$$\alpha_{i,j}^{(r)} = \frac{\exp\left(a_{i,j}^{(r)}\right)}{\sum_{r'\in\mathcal{R}} \sum_{k\in\mathcal{N}_{r'}(i)} \exp\left(a_{i,k}^{(r')}\right)} \tag{6}$$

where:

- $\mathcal{R}$ is the set of relation types.
- $\mathcal{N}_{r'}(i)$ is the set of neighbors of node i.

4.2.5 Node Update Rule

Finally, the embeddings are updated based on attention scores and original embeddings:

$$e_i^{(t+1)} = \sum_{r'\in\mathcal{R}} \sum_{k\in\mathcal{N}_{r'}(i)} \alpha_{i,k} X_k \tag{7}$$

5 Implementation Details

5.1 Training Objective

To learn contextual embeddings, we employ a masked word prediction objective. We randomly mask 15% of words and reconstruct them based on their neighboring context. The loss function used is:

$$\mathcal{L}_{\text{MSE}}(X_M^{t+1}, X_M) = \frac{\sum_{i \in M} (X_{M_{i,j}}^{t+1} - X_{M_{i,j}})^2}{m * d} \tag{8}$$

where:

- X_M^{t+1} are the predicted embeddings for masked words.
- X_M are the original embeddings of masked words.
- M is the set of masked words, and m represents its size.
- d is the embedding dimension.

5.2 Training Configuration

The model was trained for 500 epochs using the Adam optimizer with a learning rate of 0.001. A batch size of 32 was used, processing 32 sentences per iteration. The final architecture consisted of two RGAT layers: the first layer had an input dimension of 300 and an output dimension of 256, while the second layer had an input dimension of 256 and an output dimension of 300.

The decision to train for 500 epochs was based on computational constraints, ensuring that both training and evaluation could be completed within a feasible timeframe. While additional epochs may have further improved performance, empirical observations indicate that the loss plateaued after approximately 300 epochs, suggesting diminishing returns from extended training.

The training loss evolution is shown below:

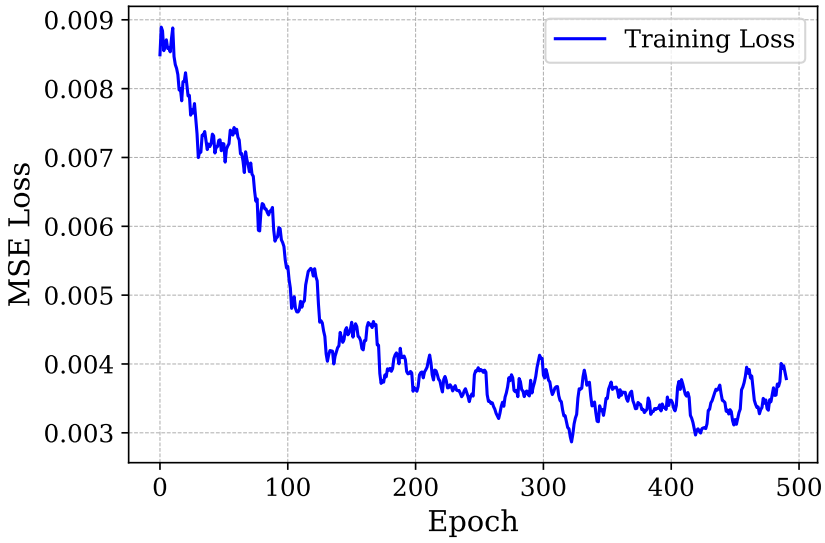


Figure 6: Masked MSE training loss over 500 epochs.

5.3 Training efficiency

To optimize memory usage and computational efficiency, only the graphs corresponding to the sentence under evaluation were utilized during each iteration of the backward pass. Although the co-occurrence graph is a single global structure, during training, we dynamically extracted the

subgraph containing only the words present in the current sentence. This approach significantly reduced memory overhead and ensured efficient computation.

With a batch size of 32, each epoch required approximately 14 seconds to process, resulting in a total training time of approximately 2 hours for 500 epochs.

All training and pre-processing were conducted exclusively on CPU resources.

5.4 Prediction Process

At inference time, the model relies on the precomputed co-occurrence matrix, ensuring consistency with training. However, this fixed structure limits the model to processing only words within the predefined vocabulary, leading to potential information loss for out-of-vocabulary (OOV) words if the vocabulary is not sufficiently comprehensive.

The syntactic and similarity graphs are constructed dynamically for each input sentence. The syntactic graph is generated using the Stanza library, while the similarity graph is built using pre-trained FastText embeddings

Once the multi-relational graph is constructed, the Relational Graph Attention Network (RGAT) processes the sentence, producing a contextualized vector embedding for each word.

6 Empirical Results

6.1 Evaluation on the WiC Dataset

To assess the quality of the contextual embeddings produced by our AGWR model, we conducted an evaluation using the Word-in-Context (WiC) dataset, a standard benchmark for word sense disambiguation. The WiC dataset provides labeled pairs of sentences containing a target word, with a binary label indicating whether the word is used with the same meaning in both sentences.

For this task, we adopted a simple classification approach based on cosine similarity. Given a pair of sentences, we computed the embeddings of the target word in each sentence using AGWR and measured their cosine similarity. A threshold-based decision rule was applied, where pairs with a similarity score above a predefined threshold were classified as having the same meaning, while those below the threshold were classified as having different meanings. While we used a naïve similarity-based approach, more sophisticated classifiers could be trained by using cosine similarity as a predictive feature and optimizing classification thresholds.

As mentioned in earlier sections,the dataset consists of 5,428 training examples and 638 validation examples. Since our AGWR model was not explicitly trained on this dataset, we treated the training set as a calibration set, selecting an optimal classification threshold, which was then applied to the test set to compute the final accuracy. The structure of the dataset and our experimental setup are illustrated in the following table:

Sentence 1	**Sentence 2**	**Target Word**	**Cosine Similarity**	**Label**
Do you want to come over to my place later?	A political system with no place for the less prominent groups	place	0.65	False
The governor should act on the new energy bill.	Think before you act.	act	0.86	True

Table 2: Example entries from the dataset used for evaluation.

To determine the optimal threshold for classification, we computed accuracy scores across different threshold values for both the training and validation sets. The results are presented below:

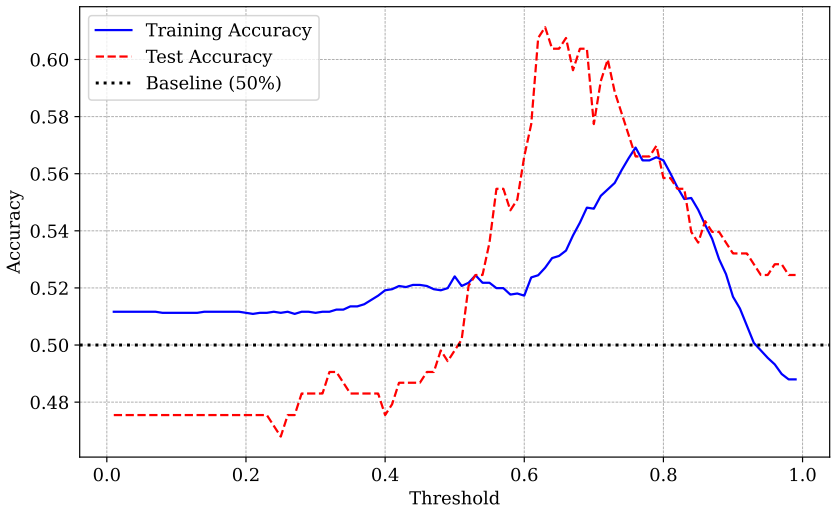


Figure 7: Train and test accuracy for different threshold values.

The accuracy on the training set ranged from approximately 52% to 57%, with the highest accuracy achieved at a similarity threshold of 0.78. The same threshold applied to the test set resulted in 57% accuracy, with a peak test set accuracy of 60% observed at a different threshold. However, since the classification threshold was selected based on the training set, we report the 57% test accuracy as our official result.

We compare our model's performance against state-of-the-art word-level contextualized embeddings from the official WiC leaderboard, as shown next:

Word-level Contextualized Embeddings	**Accuracy (%)**
WSD	67.7
BERT-large	65.5
Context2vec	59.3
ELMo	57.7
AGWR (Ours)	57.0
Random Baseline	50.0

Table 3: Comparison of word-level contextualized embeddings on the WiC dataset.

Our model achieves comparable performance to ELMo, a large-scale pre-trained model trained on approximately 1 billion words. Given that AGWR was trained on a significantly smaller corpus and with limited computational resources, these results suggest that our graph-based embeddings capture meaningful contextual information. Expanding the training corpus and increasing computational capacity could further improve performance.

6.2 Semantic Clustering

To analyze whether our embeddings effectively distinguish different word senses, we performed an additional evaluation using a synthetic dataset specifically designed to test semantic clustering. We constructed a dataset containing 100 sentences, each including the word "bank," with 50 instances referring to a financial institution and 50 referring to the edge of a river. These sentences were generated using Large Language Models (LLMs).

To visualize the learned representations, we applied t-distributed Stochastic Neighbor Embedding (t-SNE) [19] to reduce the 300-dimensional embeddings into a two-dimensional space. The results are presented next:

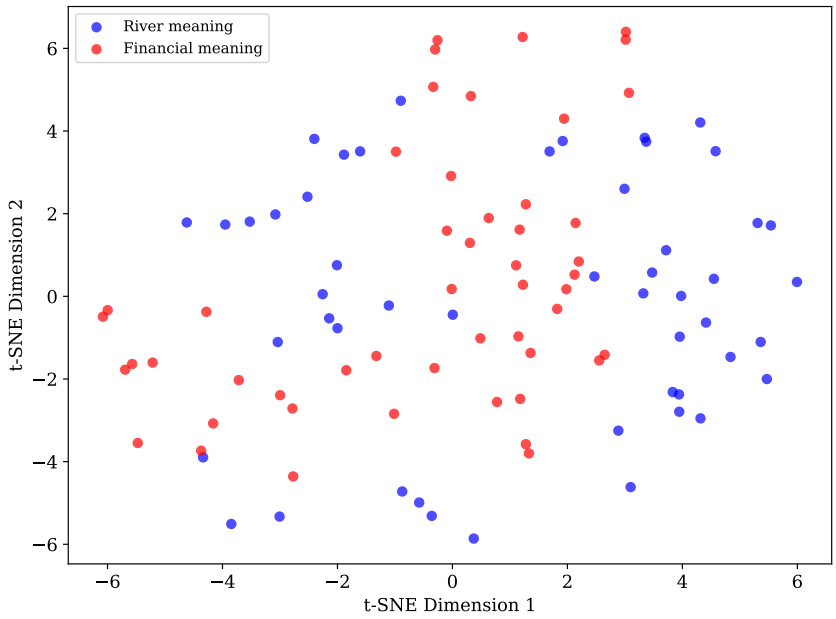


Figure 8: 2D t-SNE visualization of AGWR embeddings for sentences containing "bank" in different contexts.

The visualization reveals clear clustering patterns. Words associated with the financial sense of "bank" are predominantly concentrated in the left region, while those referring to the riverbank meaning appear in the right region. The first t-SNE dimension appears to be the most discriminative, separating the two clusters along the x-axis.

To quantify this observation, we partitioned the data into three regions based on t-SNE Dimension 1:

1. Region 1 (t-SNE dim > 2): 76% of sentences refer to "bank" as a riverbank.
2. Region 2 (-1 < t-SNE dim < 2): 57% refer to "bank" as a financial institution.
3. Region 3 (t-SNE dim < -1): 56% refer to "bank" as a financial institution

Compared to a random baseline of 50%, these results confirm that AGWR embeddings successfully capture semantic distinctions between polysemous words, allowing for effective differentiation of contextual meanings.

7 Discussion

The results demonstrate that AGWR embeddings effectively capture contextual word relationships and meaning distinctions despite being trained on a limited corpus with constrained computational resources. Performance on the WiC dataset is competitive with ELMo, and the t-SNE clustering results provide further evidence that AGWR embeddings encode semantically meaningful representations.

However, while our current results are promising, further improvements could be achieved.

7.1 Limitations and Future Directions

While the integration of syntactic, similarity, and co-occurrence relationships has demonstrated the potential of graph-based contextual embeddings, the results obtained in this study do not yet fully capture the capabilities of these models. Several limitations in our implementation hindered

optimal performance, and addressing these challenges could lead to significant improvements in future iterations.

7.1.1 Computational Constraints and Vocabulary Size

One of the primary limitations of our implementation was the restricted computational resources, which constrained the size of the processed corpus and the overall vocabulary. Our analysis was limited to a vocabulary of 25,000 words, meaning that many words present in the training and test sentences were excluded. This resulted in information loss during both training and prediction, as key words in a given context were sometimes omitted. For instance, the sentence *I play pickleball* would be processed as *I play* because *pickleball* is not included in the vocabulary. Expanding the corpus and leveraging more computational resources would allow for a larger and more representative vocabulary, improving contextual learning.

7.1.2 Hyperparameter Optimization and Training Time

Due to time constraints, extensive hyperparameter tuning and architectural exploration were not conducted. Our model was trained for 500 epochs without systematic optimization of parameters such as the number of layers, attention heads, or learning rate. Future iterations could benefit from a more rigorous hyperparameter search. Additionally, training for more epochs could help identify an optimal stopping point for better generalization.

7.1.3 Vocabulary Coverage in the WiC Dataset

While our model was tested on the WiC dataset, our ability to compute contextual embeddings was limited to words that existed in our predefined vocabulary. Although the majority of words in WiC could be processed, some were excluded, potentially affecting evaluation accuracy. Tokenization-based strategies could be explored in the future to mitigate this issue.

7.1.4 Dependency on External Syntactic Parsers

At inference time, our model relies on syntactic parsers, such as the Stanza library, to generate dependency graphs. While modern NLP tools provide accurate parsing, a fully self-contained approach could involve predicting syntactic relationships using only co-occurrence and similarity graphs rather than relying on an external parser. Exploring this alternative could reduce reliance on pre-trained parsing models.

7.1.5 Supervised Learning and Alternative Loss Functions

Our current approach focuses on semi-supervised learning through masked word prediction. However, given the availability of labeled data in the WiC dataset, an alternative would be to train the model in a supervised manner using a contrastive loss function.

Contrastive loss is designed to bring embeddings of semantically similar words closer together while pushing apart embeddings of words used in different contexts. Given a pair of word embeddings $(\mathbf{x}_i, \mathbf{x}_j)$ and a binary label y_{ij} indicating whether the words share the same meaning ($y_{ij} = 1$) or not ($y_{ij} = 0$), the contrastive loss function is defined as:

$$\mathcal{L}_{\text{contrastive}} = (1 - y_{ij}) \max(0, m - d_{ij})^2 + y_{ij} d_{ij}^2$$

where:

- $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ is the Euclidean distance between the two embeddings.
- m is a margin hyperparameter that sets the minimum separation distance for negative pairs.
- If $y_{ij} = 1$, meaning the words are used with the same meaning, the loss encourages their embeddings to be close (i.e., minimizes d_{ij}^2).
- If $y_{ij} = 0$, meaning the words have different meanings, the loss enforces a minimum separation by penalizing distances smaller than m.

7.2 Potential Extensions and Additional Experiments

Beyond improving the existing framework, several additional tests could be conducted to further evaluate and enhance the quality of our contextual embeddings.

7.2.1 Application to Downstream Tasks

One direct application of our learned embeddings is their integration into downstream NLP tasks, such as next-word prediction, sentiment analysis, or machine translation. Evaluating AGWR embeddings in these tasks could provide deeper insights into their generalization ability and practical utility.

7.2.2 Enhanced Classification in WiC

While our current evaluation in the WiC dataset is based on a thresholded cosine similarity, a more sophisticated classification model could be trained.

7.2.3 Scaling the Clustering Task

The clustering evaluation of the word *bank* provided an initial indication of semantic separation, but the dataset size was relatively small. Increasing the number of sentences per word sense and applying unsupervised clustering algorithms such as K-means could lead to a more robust analysis.

8 Conclusion

This study explored the integration of co-occurrence, syntactic, and similarity-based relationships in learning contextual word embeddings using graph neural networks. Our Adaptive Graph-Based Word Representations (AGWR) demonstrated promising results despite being trained on a limited corpus with low computational resources. These findings suggest that incorporating syntactic relationships into the learning process enhances the semantic understanding of words, addressing some of the limitations inherent in traditional co-occurrence-based embeddings. While co-occurrence graphs have been extensively studied in the literature, their constraints are well-documented. Few studies have explored syntactic relationships, and even fewer have considered the integration of multiple relational signals into a unified graph-based framework.

The multi-relational GNN approach introduced in this work efficiently captures structured word relationships while remaining computationally tractable. Our model achieved competitive performance in the WiC benchmark and exhibited strong semantic clustering capabilities, indicating its potential as an alternative to transformer-based models. However, several limitations, including restricted vocabulary size, reliance on syntactic parsers, and lack of extensive hyperparameter tuning, highlight opportunities for further optimization.

Future work will focus on expanding the corpus size, refining training procedures, and exploring supervised learning paradigms such as contrastive learning to improve word sense differentiation. Additionally, integrating AGWR embeddings into downstream NLP tasks like sentiment analysis, machine translation, and information retrieval could further validate their effectiveness. By continuing to refine these techniques, graph-based word representations could offer a scalable, interpretable, and resource-efficient alternative to conventional sequential models, contributing to advancements in contextual word representation learning in NLP.

References

[1] Ragheb Al-Ghezi and Mikko Kurimo. Graph-based syntactic word embeddings. In Dmitry Ustalov, Swapna Somasundaran, Alexander Panchenko, Fragkiskos D. Malliaros, Ioana Hulpuș, Peter Jansen, and Abhik Jana, editors, *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, pages 72–78, Barcelona, Spain (Online), December 2020. Association for Computational Linguistics.

[2] Mikhail Belkin and John Goldsmith. Using eigenvectors of the bigram graph to infer morpheme identity, 2002.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2017.

[4] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y. Hammerla. Relational graph attention networks, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[6] Xingzhi Guo and Steven Skiena. Hierarchies over vector space: Orienting word and graph embeddings, 2024.

[7] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson, 2019. Draft of October 2, 2019.

[8] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[9] Wenting Li, Jiahong Xue, Xi Zhang, Huacan Chen, Zeyu Chen, Feijuan Huang, and Yuanzhe Cai. Word-graph2vec: An efficient word embedding approach on word co-occurrence graph using random walk technique, 2023.

[10] Ziran Liang, Yuyin Lu, HeGang Chen, and Yanghui Rao. Graph-based relation mining for context-free out-of-vocabulary word embedding learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14133–14149, Toronto, Canada, July 2023. Association for Computational Linguistics.

[11] Yuyin Lu, Xin Cheng, Ziran Liang, and Yanghui Rao. Graph-based dynamic word embeddings. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4280–4288. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[13] Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. Universal Dependencies. In Alexandre Klementiev and Lucia Specia, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, Valencia, Spain, April 2017. Association for Computational Linguistics.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[15] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[16] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A python natural language processing toolkit for many human languages, 2020.

[17] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. OpenAI blog, 2019.

[18] Max Ryabinin, Sergei Popov, Liudmila Prokhorenkova, and Elena Voita. Embedding words in non-vector space with unsupervised graph learning, 2020.

[19] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[20] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[21] Väinö Yrjänäinen and Måns Magnusson. Probabilistic embeddings with laplacian graph priors, 2022.

A Repository Overview and Reproduction Guidelines

This appendix provides an overview of the repository structure, data processing, training, testing, and prediction steps for the project. Additionally, guidelines to reproduce the results are included.

A.1 Repository Structure

The repository is organized as follows:

- `utils/`: Contains utility functions and classes.
- `pipelines/`: Contains scripts for data processing, training, and testing.
  - `data_processing/`: Scripts for data processing.
  - `train/`: Scripts for training the model.
  - `test/`: Scripts for testing and prediction.
- `objects/`: Directory for storing models, graphs, and other artifacts.
- `parameters/`: Directory containing parameter configuration files.

A.2 Data Processing

The data processing steps are implemented in the `data_processing` scripts. The main functions include:

- `preprocess_words`: Preprocesses words by converting them to lowercase and removing punctuation.
- `get_top_k_words_brown`: Retrieves the top K most frequent words from the Brown corpus.
- `preprocess_sentences`: Preprocesses sentences by converting words to lowercase and removing punctuation.
- `generate_syntactic_graphs`: Generates syntactic graphs from sentences.
- `generate_cooccurrence_graph`: Generates a co-occurrence graph from sentences.
- `generate_similarity_graphs`: Generates similarity graphs from sentences.
- `filter_graphs`: Filters out graphs with missing nodes or empty edges.

A.3 Training

The training process is implemented in the `train` scripts. The main functions include:

- `train_model`: Trains the ContextRGAT model.
- `save_checkpoint`: Saves the model checkpoint.
- `load_checkpoint`: Loads the model checkpoint.

A.4 Testing and Prediction

The testing and prediction steps are implemented in the `test` scripts. The main functions include:

- `process_dataset`: Processes the dataset and generates predictions.
- `predict_embeddings`: Predicts embeddings for sentences using the trained model.

A.5 Reproduction Guidelines

To reproduce the results, follow these steps:

1. Clone the repository:

```
git clone <repository_url>
cd <repository_directory>
```

2. Install the required dependencies:

```
pip install -r requirements.txt
```

3. Download cc.en.300.bin from FastText and save it in the `utils/fasttext`:
4. Preprocess the data:

```
python pipelines/data_processing/data_processing_brown.py
```

5. Train the model:

```
python pipelines/train/brown/training_brown.py
```

6. Test the model and generate predictions:

```
python pipelines/test/test_brown_wic.py
```

7. Resume training from a checkpoint (if needed):

```
python pipelines/train/brown/resume_training_brown.py
```

A.6 Additional Details

- Ensure that the paths in the scripts are correctly set to point to the appropriate directories and files.
- The `parameters.yaml` file contains configuration parameters such as the number of epochs, batch size, learning rate, and checkpoint interval. Modify these parameters as needed.
- The `objects/` directory is used to store models, graphs, and other artifacts generated during the data processing, training, and testing steps.

By following these guidelines, you should be able to reproduce the results and further experiment with the model and data.