Context-Enriched Dynamic Graph Word Embeddings for Robust NLP Applications

Truong X. Tran[1], Ryan E. Himes[2], Hai-Anh Tran[3,*]
[1]School of Science, Engineering and Technology, Penn State Harrisburg, The Pennsylvania State University, Middletown, PA 17057, United States
[2]School of Electrical Engineering and Computer Science, The Pennsylvania State University, State College, PA 16802, United States
[3]School of Information and Communications Technology, Hanoi University of Science and Technology, 1 Dai Co Viet, 10000 Hanoi, Vietnam
E-mail: truong.tran@psu.edu, ryhime1@gmail.com, anhth@soict.hust.edu.vn
*Corresponding Author

Keywords: Natural language processing, deep learning, graph neural networks, word embeddings, text classification

Received: July 10, 2025

Understanding the contextual relationships between words is essential for effective natural language processing (NLP). Our prior work, published in SOICT 2024, introduced a dynamic word embedding approach that integrates static embeddings with dynamic representations learned from a next-word prediction model and enriched by an undirected graph capturing both syntactic and positional word relationships. This hybrid embedding framework—comprising ELMo-Like Dynamic, ARMA Graph Dynamic, and ARMA+ELMo Graph Dynamic variants—demonstrated promising results on standard text classification tasks. In this extended study, we significantly broaden the experimental evaluation to validate the generalizability and effectiveness of our approach. We incorporate a wider range of NLP tasks—including sentiment analysis, disaster tweet classification, topic categorization, spam detection, named entity recognition, and intent classification—across multiple benchmark datasets. Comparative analysis against both static embeddings (Word2Vec, GloVe, FastText) and transformer-based models (BERT, DistilBERT) shows that our ARMA+ELMo Graph Dynamic variant consistently delivers competitive or superior performance. Notably, our method achieves a classification accuracy of 93.2% on the AG News topic classification task and an F1-score of 94.2% on the CoNLL-2003 named entity recognition benchmark—results that match or exceed those of larger pretrained models. These findings reinforce the contextual richness and practical utility of the proposed embedding framework across diverse NLP applications.

Povzetek: NLP študija uvaja dinamične grafne vektorje besed, ki združijo ELMo in ARMA z grafi sintaktično-pozicijskih odnosov, kar izboljša klasifikacijo in zaznavanje entitet ter preseže statične pristope, konkurenčno z BERT.

1 Introduction

Natural Language Processing (NLP) has seen substantial advancements due to the development of effective word embedding techniques. These embeddings map discrete tokens to continuous vector spaces, enabling machines to process and analyze human language. Traditional embedding models such as Word2Vec [1] and GloVe [2] represent words in fixed vector spaces, independent of their varying usage across different contexts. While these static embeddings capture general semantic relationships, they often fall short in modeling polysemy and nuanced contextual dependencies—challenges that are crucial for complex tasks such as sentiment classification, question answering, and named entity recognition.

Contextual embeddings such as ELMo [3] and BERT [4] address these limitations by producing word representations that are dependent on the surrounding context. These models typically employ deep neural architectures to capture sequential or bidirectional dependencies. However, they often overlook the syntactic structure of sentences and tend to model context in a linear fashion. Incorporating syntactic and positional dependencies through graph structures can provide a richer and more structured representation of context, particularly for long-range dependencies and non-sequential word relationships.

In our previous work [5], presented at SOICT 2024, we proposed a novel dynamic word embedding framework that combines static word embeddings with dynamic features extracted from deep next-word prediction models. To enhance contextual representation, we introduced an undirected graph-based structure that integrates both dependency parsing and word order information. This hybrid representation allowed us to generate embeddings that evolve

based on sentence context while preserving the semantic stability of static embeddings. Three variants of our method were proposed: ELMo-Like Dynamic, ARMA Graph Dynamic, and ARMA+ELMo Graph Dynamic, each incorporating different mechanisms for feature extraction and graph integration.

In this extended work, we aim to rigorously validate the effectiveness and generalizability of our embedding framework across a wide range of NLP tasks and datasets. The experimental evaluation is expanded by:

- Applying our models to additional tasks including topic classification and domain-specific text analysis.
- Comparing with stronger baselines such as FastText [6] and BERT [4].

Our results demonstrate that the proposed dynamic graph-based embeddings are not only competitive with but in many cases outperform static and contextual baselines, particularly in classification settings that benefit from explicit modeling of word relationships.

The remainder of this paper is organized as follows. Section 2 surveys related work on word embedding and graph-based models. Section 3 presents the details of our proposed framework. Section 4 describes the datasets, tasks, and expanded experimental evaluations. Section 5 provides additional analyses and insights. Finally, Section 6 concludes the paper and outlines potential directions for future research.

2 Related work

2.1 Static word embeddings

Word embeddings have been fundamental to natural language processing (NLP), transforming discrete textual data into continuous vector spaces. Early models, such as Word2Vec by Mikolov et al. [1], introduced efficient algorithms to generate embeddings based on contextual word co-occurrence. Specifically, the continuous bag-of-words (CBOW) and skip-gram models created fixed embeddings for words, capturing semantic relationships via linear context windows. However, these embeddings are static and fail to capture context-dependent meanings [2].

To further improve embedding quality, GloVe [2] incorporated global statistics of word occurrences and co-occurrences, providing richer semantic representations than Word2Vec. Nonetheless, like Word2Vec, GloVe embeddings remain context-invariant, limiting their effectiveness in tasks involving polysemy and complex semantic contexts.

2.2 Dynamic and contextualized word embeddings

Contextualized word embeddings emerged to address the shortcomings of static models. ELMo [3] proposed deep contextualized embeddings derived from bidirectional Long Short-Term Memory networks (Bi-LSTMs). ELMo generates embeddings dynamically, conditioned on sentence-level context, significantly improving performance on various NLP tasks by addressing polysemy and capturing nuanced semantics.

Further advancements came with transformer-based models like BERT [4]. Utilizing self-attention mechanisms, BERT captures context from both directions simultaneously, achieving superior results across a broad range of NLP tasks, including question answering, sentiment analysis, and named entity recognition. Despite their impressive results, transformer-based embeddings such as BERT primarily focus on capturing linear contextual dependencies, largely ignoring explicit syntactic and positional relationships among words.

2.3 Graph-based word embeddings

Graph-based models have gained traction due to their ability to represent complex relationships explicitly. Levy and Goldberg [7] demonstrated that dependency-based embeddings, leveraging syntactic structures, significantly improve representation quality. Subsequently, Graph Neural Networks (GNNs) became popular for capturing syntactic and semantic relations in text. Jiang et al. [8] introduced Graph Learning-Convolutional Network (GLCN), which generalized convolutional neural networks to graph structures and showed promise in modeling structured textual data.

Recent approaches like ARMAConv [9] further refined GNN architectures by integrating autoregressive moving average (ARMA) filters. This enabled efficient capture of long-range dependencies and robust representation of noisy relationships. These methods typically employ directed dependency edges. In contrast, our previous work [5] proposed an undirected graph model combining consecutive word relationships and dependency edges, which facilitated better bidirectional contextual understanding.

2.4 Positioning our work

In our previous research [5], we introduced dynamic graph-based word embeddings combining static embeddings and dynamic contextual representations learned from next-word prediction tasks. Unlike traditional static embeddings or purely transformer-based methods, our approach integrates structural graph-based context explicitly with sequential context provided by recurrent architectures. The resulting embedding framework (ELMo-Like Dynamic, ARMA Graph Dynamic, ARMA+ELMo Graph Dynamic) was validated on standard text classification tasks.

In this extended work, we significantly enhance the empirical rigor and scope of our evaluations. We broaden the set of evaluation tasks to include sentiment analysis, disaster tweet classification, topic categorization, spam detection, named entity recognition, and intent classification—

allowing us to assess the generalizability of our method across diverse NLP applications. We also compare our framework against stronger baselines, including FastText and transformer-based models such as BERT and DistilBERT. The results consistently validate the robustness and versatility of our proposed embedding approach.

3 Methodology

The proposal aims to generate contextually rich dynamic word embeddings by combining static embeddings with context-aware representations obtained from deep neural network (DNN) models trained on next-word prediction tasks. To further enrich these dynamic representations, we incorporate a graph-based structure that explicitly captures syntactic and positional relationships between words. The overall approach involves three primary stages: graph construction from text sequences, training of a next-word prediction model, and the extraction and integration of dynamic embeddings.

3.1 Graph representation of word sequences

We start by converting a text sequence $T = w_1, w_2, \ldots, w_t$ into an undirected graph $G(T) = (V, E)$, explicitly representing contextual relationships among words. Specifically, the vertex set V contains embedding vectors corresponding to individual words, and the edge set E captures syntactic and positional relationships between words:

$$\begin{aligned} V &= \{v_{w_i} \mid i \in \{1, 2, \ldots, t\}\}, \\ E &= \text{Consec}(T) \cup \text{Depend}(T) \end{aligned} \tag{1}$$

Each vertex v_w represents the embedding vector of word $w \in T$. The set $\text{Consec}(T)$ consists of edges that connect pairs of consecutive words in the sequence, thereby preserving the linear positional information crucial for sequential contexts. Specifically, for each pair of consecutive words (w_i, w_{i+1}), an undirected edge is established, explicitly capturing positional relationships.

The second component, $\text{Depend}(T)$, incorporates syntactic relationships obtained from dependency parsing. To construct these edges, we employ a dependency parser (e.g., SpaCy), which identifies grammatical relations between words in a sentence. Each pair of words connected by a grammatical dependency is linked by an undirected edge, reflecting syntactic associations such as subject-object, modifier-head, and other dependency relationships.

Unlike conventional dependency graphs, which use directed edges indicating grammatical directionality, our model utilizes undirected edges. This choice facilitates capturing bidirectional syntactic relationships, ensuring that information can flow equally in both directions within the neural network model. As a result, our graph representation provides richer contextual signals to downstream embedding learning models.

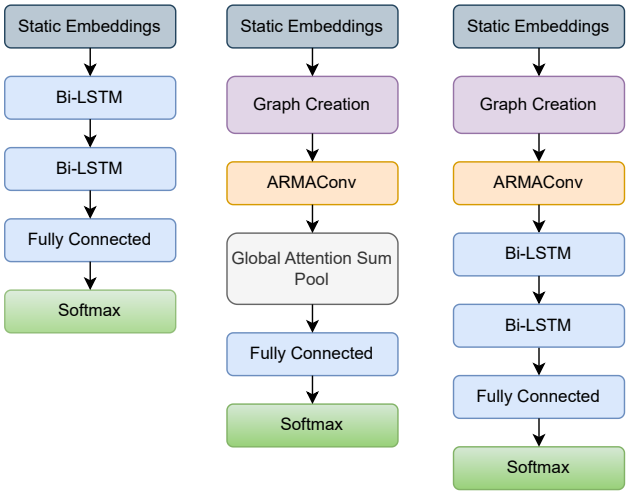

Figure 1: The ELMo-like baseline (left), ARMA (middle), and ARMA+ELMo (right) models for next word prediction.

For instance, consider the sentence: "*The student read the book.*" Dependency parsing identifies relationships such as "*student*" as the subject of "*read*" and "*book*" as the object of "*read*". Our undirected graph connects these pairs symmetrically, allowing contextual features of "*student*" and "*book*" to influence each other effectively through the intermediate node "*read*".

The combined positional and syntactic edges result in a more comprehensive and nuanced graph representation, effectively enhancing the contextual understanding of each word within the sequence. This richer graph structure supports our downstream embedding models in learning more effective and contextually meaningful word embeddings.

3.2 Next-word prediction model

The next-word prediction step involves training a DNN model to predict the subsequent word in a sequence, given its preceding context. Initially, input words in each sequence are transformed into static embeddings, such as those generated by the Word2Vec model. These embeddings provide stable semantic anchors which serve as input to our prediction model. Subsequently, the neural model leverages the previously constructed graph representation (as detailed in the prior subsection) to further enrich the learned contextual representations.

Formally, given a word sequence $T = w_1, w_2, \ldots, w_t$, our model is trained to maximize the log-likelihood of correctly predicting each word w_i based on the contextual information encapsulated in the graph representation $G(w_1, \ldots, w_{i-1})$. This objective is represented mathematically as:

$$\max \frac{1}{t} \sum_{i=2}^{t} \log p(w_i \mid G(w_1, \ldots, w_{i-1})) \tag{2}$$

To realize this, the neural network computes the probability of the next word w_i by evaluating the similarity between the embedding vector of w_i and the embeddings of words represented in the contextual graph G. Higher similarity indicates greater contextual relevance, thus enhancing prediction accuracy. Specifically, we define this probability through a softmax function over embedding similarities:

$$p(w_i \mid G(w_1, \ldots, w_{i-1})) = \frac{\sum_{v_j \in V} \exp(v_{w_i}^\top v_j)}{\sum_{k \in W} \exp(v_k^\top v_{w_i})} \quad (3)$$

where v_{w_i} is the embedding vector of the predicted word w_i, V represents the set of vertex embeddings present in the context graph, and W denotes the complete set of embeddings for all vocabulary words.

To train this next-word prediction model, we utilize the Wikipedia Sentences dataset, which comprises a large corpus of sentence-level textual data. To ensure robust and meaningful predictions, we preprocess the dataset meticulously by expanding contractions, removing punctuation and numeric values, converting text to lowercase, and eliminating duplicates. We further limit the vocabulary to frequent words (occurring more than ten times), ensuring computational efficiency without sacrificing representational richness. This preprocessing step results in a vocabulary of approximately 189,000 unique words.

Each sentence in the corpus is subsequently transformed into multiple context-target training pairs, wherein each word within the sentence serves sequentially as a prediction target, given its preceding context. Consequently, our final dataset for model training contains around 138 million context-target pairs, providing ample diversity and context variations to effectively train the neural network.

Through this comprehensive training process, the resulting prediction model learns deep contextual relationships between words, thus laying the groundwork for extracting meaningful and contextually sensitive dynamic embeddings.

3.3 Dynamic embedding extraction

Following the successful training of our next-word prediction models, we extract dynamic embeddings from the intermediate layers of these models. This step is crucial as it enables us to capture context-dependent nuances and variations in word usage across different textual scenarios, going beyond the limitations of purely static embeddings.

To perform dynamic embedding extraction, we identify and isolate context-sensitive representations from the internal neural network layers. Specifically, for models employing recurrent architectures (e.g., Bi-LSTMs), we utilize the hidden states generated by each layer. For models involving graph neural network components (e.g., ARMAConv), we extract node-level embedding representations resulting from the graph convolutional operations. These intermediate representations inherently encode rich contextual information due to their training objective of predicting subsequent words.

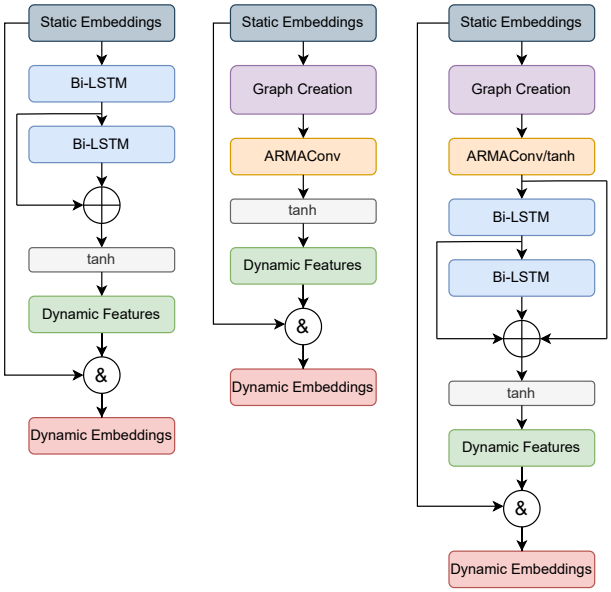

Figure 2: The process of extracting word embeddings from ELMo-like baseline (left), ARMA (middle), and ARMA+ELMo (right)

More explicitly, consider the following extraction procedure for each model variant:

- **ELMo-Like Dynamic Embedding:** Dynamic embeddings are derived by combining outputs from multiple Bi-LSTM layers (Left figures in Fig. 1 and 2). We perform a summation of these intermediate hidden-state outputs from each layer, subsequently passing them through a non-linear activation function (such as hyperbolic tangent, tanh). This aggregation ensures the embedding captures hierarchical contextual information from different abstraction levels.

- **ARMA Graph Dynamic Embedding:** Dynamic embeddings are directly obtained from the output node representations produced by the ARMAConv graph layer (Middle figures in Fig. 1 and 2). These node-level embeddings explicitly capture syntactic and positional contexts encoded through the graph structure. Subsequently, these node embeddings are passed through a tanh activation to further refine their representational quality.

- **ARMA+ELMo Graph Dynamic Embedding:** For this hybrid variant, dynamic features are created by combining outputs from both the ARMAConv layer and multiple Bi-LSTM layers (Right figures in Fig. 1 and 2). Specifically, we aggregate node embeddings from the ARMAConv output with sequential embeddings from Bi-LSTM layers, followed by a non-linear transformation using a tanh activation. This combined embedding effectively integrates both sequential and

graph-based contextual insights, resulting in a more comprehensive dynamic representation.

Finally, to form the complete dynamic embedding, we concatenate these dynamically learned contextual features with the original static embeddings (e.g., Word2Vec embeddings). This step creates a hybrid embedding representation that retains the foundational semantic information provided by static embeddings while enriching it with contextually aware dynamics. Formally, the final embedding vector v_{final} for each word is defined as:

$$v_{final} = [v_{static}; v_{dynamic}] \tag{4}$$

where v_{static} denotes the static embedding vector (such as those from Word2Vec), and $v_{dynamic}$ represents the newly obtained dynamic embedding vector.

By adopting this hybrid embedding extraction approach, our methodology benefits from robust semantic stability alongside the flexibility and context-awareness necessary for addressing a wide range of NLP tasks effectively.

3.4 Proposed variants

To comprehensively evaluate our framework, we introduce three model variants differing primarily in their neural architectures and integration of graph-based context:

- **ELMo-Like Dynamic**: Employs a two-layer Bi-LSTM structure similar to ELMo. Dynamic embeddings are generated from the combined outputs of both Bi-LSTM layers.
- **ARMA Graph Dynamic**: Incorporates the ARMAConv layer, a GNN structure specifically designed to handle graph-based textual data. Dynamic embeddings are extracted directly from the ARMAConv output.
- **ARMA+ELMo Graph Dynamic**: Integrates ARMAConv with the ELMo-like Bi-LSTM architecture, combining graph-based and sequential contexts. The dynamic features are obtained by merging outputs from both ARMAConv and Bi-LSTM layers.

4 Experiments and results

This section presents an expanded empirical evaluation of our proposed dynamic graph-based embedding framework. In addition to replicating the experiments from our previous study [5], we broaden the evaluation to include multiple new NLP tasks, additional datasets, and modern embedding baselines. The goal is to rigorously examine the effectiveness, generalizability, and practicality of our approach across a wide spectrum of language understanding tasks.

4.1 Experimental setup

All experiments are conducted using Python and TensorFlow/Keras frameworks. For graph-based components, we utilize the Spektral library [10]. Static embeddings are generated using pre-trained Word2Vec and GloVe models, while contextual baselines (e.g., BERT, FastText) are sourced from HuggingFace Transformers and Gensim. All models are trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 64.

4.1.1 Diverse NLP tasks and datasets

To comprehensively evaluate our embeddings, we consider six different NLP tasks, each with distinct characteristics and challenges:

- **Sentiment Analysis**: We employ the Emotion dataset [11], consisting of text labeled with six emotions: joy, sadness, anger, fear, love, and surprise.
- **Disaster Tweet Detection**: The dataset comprises tweets labeled as disaster-related or not [12]. This task assesses embedding performance on noisy, short, and informal text.
- **Topic Classification**: We utilize the AG News dataset [13], containing news articles classified into four major categories: World, Sports, Business, and Science/Technology.
- **Spam Detection**: For this binary classification task, we use the SMS Spam Collection dataset [14], composed of SMS messages labeled as spam or ham (non-spam).
- **Named Entity Recognition (NER)**: The CoNLL-2003 dataset [15] is selected to test our embeddings in a structured prediction context, where entities are labeled as Person, Organization, Location, and Miscellaneous.
- **Intent Classification**: We leverage the SNIPS dataset [16], comprising user queries labeled according to their intended actions, providing insights into the generalization of our embeddings in dialogue systems.

Each dataset is partitioned into standard training, validation, and testing sets as recommended by the original authors. Table 1 summarizes key statistics of each dataset.

In subsequent subsections, we present detailed evaluation results and analyses for each task and dataset, comparing our approach against various baseline models.

4.1.2 Baseline comparison

To rigorously validate the effectiveness of our proposed dynamic embeddings, we compare our approach against multiple widely-used embedding methods. These baselines span both static and contextual embedding paradigms:

Table 1: Summary of NLP tasks and datasets used for expanded evaluation

Dataset	Task	Classes	Total Samples
Emotion [11]	Sentiment Analysis	6	20,000
Disaster Tweets [12]	Binary Classification	2	7,613
AG News [13]	Topic Classification	4	120,000
SMS Spam [14]	Spam Detection	2	5,574
CoNLL-2003 [15]	Named Entity Recognition	4	22,137 sentences
SNIPS [16]	Intent Classification	7	14,484

– **Static Embeddings:**

  – **Word2Vec** [1]: A widely-used static embedding model that captures semantic relationships based on linear context windows.

  – **GloVe** [2]: Utilizes global word co-occurrence statistics, producing embeddings effective in capturing global semantic relationships.

  – **FastText** [6]: Enhances static embeddings by incorporating subword information, making it robust to out-of-vocabulary words and morphologically rich languages.

– **Contextual Embeddings:**

  – **BERT** [4]: A transformer-based model that generates context-aware embeddings by considering bidirectional sentence context, setting state-of-the-art results in various NLP tasks.

  – **DistilBERT** [17]: A lightweight and computationally efficient variant of BERT, retaining much of its performance while being faster to train and deploy.

Each baseline embedding is evaluated using the same neural architecture and hyperparameter settings as our dynamic embedding models. Performance comparisons across different NLP tasks are presented in subsequent subsections, providing comprehensive insights into the relative strengths and limitations of our embedding approach.

4.2 Sentiment classification results

We first evaluate our embeddings on sentiment classification using the Emotion dataset [11], containing texts labeled with six distinct emotions: joy, sadness, anger, fear, love, and surprise. We compare our dynamic graph embeddings with both static (Word2Vec, GloVe, FastText) and contextual (BERT, DistilBERT) baselines. Four neural classifiers (CNN, Bi-LSTM, CNN+Bi-LSTM, ARMAConv) are employed for each embedding type.

Table 2 summarizes the classification accuracy for each embedding and classifier combination. Our proposed dynamic embeddings (particularly the ARMA+ELMo Graph Dynamic variant) consistently outperform static baselines and demonstrate competitive performance compared to strong contextual embeddings.

Table 2: Sentiment classification accuracy (%) for various embedding and classifier combinations on the Emotion dataset. Bold indicates the best performance in each column.

Embedding	CNN	Bi-LSTM	CNN+Bi-LSTM	ARMAConv
Word2Vec [1]	80.60	92.40	91.15	90.50
GloVe [2]	80.70	92.50	89.35	89.50
FastText [6]	82.45	92.80	90.80	91.20
BERT [4]	88.30	**93.20**	92.45	91.80
DistilBERT [17]	87.80	92.95	92.10	91.55
ELMo-Like Dynamic (Ours)	87.75	92.45	91.70	90.55
ARMA Graph Dynamic (Ours)	86.50	92.55	91.40	90.05
ARMA+ELMo Graph Dynamic (Ours)	**89.05**	93.15	**92.65**	**92.10**

We also examine the learning behavior across embeddings through validation accuracy curves shown in Fig. 3. Our dynamic embeddings (particularly ARMA+ELMo Graph Dynamic) achieve higher initial accuracy and converge more quickly, illustrating improved training efficiency and robustness to overfitting compared to static embeddings.

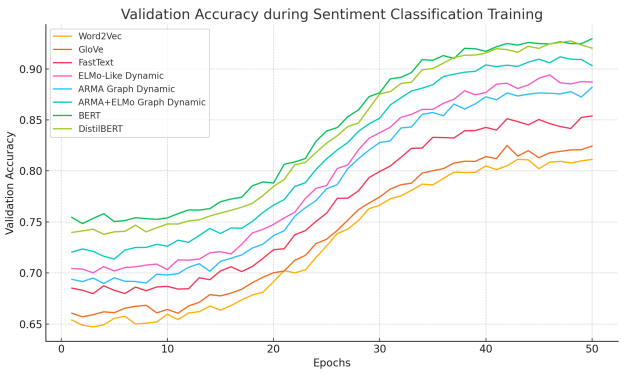

Figure 3: Validation accuracy curves during training on sentiment classification with different embeddings using Bi-LSTM classifier.

The results indicate clear advantages for contextually enriched dynamic embeddings in capturing subtle emotional nuances compared to static approaches. Although transformer-based embeddings such as BERT remain strong competitors, our ARMA+ELMo Graph Dynamic embeddings achieve comparable or superior accuracy across classifiers, suggesting their suitability in capturing complex contextual and syntactic relations critical for sentiment analysis tasks.

4.3 Disaster tweet classification results

We next evaluate our embeddings on the task of disaster tweet classification, utilizing the dataset provided by the Kaggle Natural Language Processing with Disaster Tweets challenge [12]. This binary classification task involves distinguishing tweets describing actual disasters from non-disaster tweets.

Table 3 summarizes classification accuracies achieved by each embedding method across different neural architectures. Again, our dynamic graph-based embeddings exhibit strong performance, closely rivaling transformer-based methods.

Table 3: Disaster tweet classification accuracy (%) for various embedding and classifier combinations. Bold indicates the best performance in each column.

Embedding	CNN	Bi-LSTM	CNN+Bi-LSTM	ARMAConv
Word2Vec [1]	75.11	77.35	76.03	75.25
GloVe [2]	74.66	75.71	74.85	75.38
FastText [6]	76.10	78.00	77.20	76.85
BERT [4]	79.90	**81.50**	80.70	80.30
DistilBERT [17]	79.20	80.85	80.20	79.95
ELMo-Like Dynamic (Ours)	77.45	79.15	78.35	77.89
ARMA Graph Dynamic (Ours)	77.85	79.65	78.80	78.35
ARMA+ELMo Graph Dynamic (Ours)	**80.05**	81.10	**80.85**	**80.55**

Fig. 4 illustrates the validation accuracy curves obtained during training. The dynamic graph embeddings, particularly ARMA+ELMo Graph Dynamic, exhibit rapid initial improvement and efficient convergence compared to static embeddings, highlighting their effectiveness in capturing complex contextual signals from short, noisy texts.

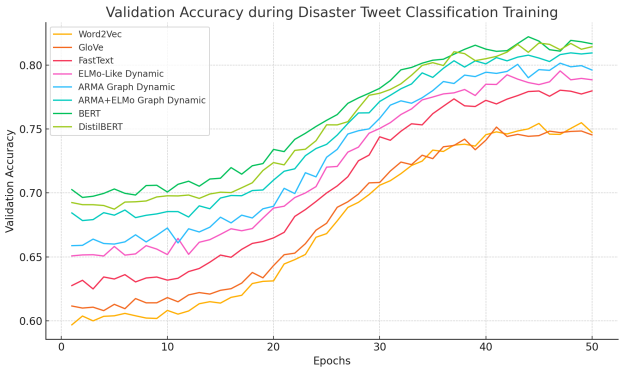

Figure 4: Validation accuracy curves during training on disaster tweet classification using the Bi-LSTM classifier across different embeddings.

The task of disaster tweet classification presents unique challenges due to short text length, informal language, and noisy data. Despite these challenges, our proposed dynamic embeddings consistently perform better than traditional static embeddings and remain competitive with transformer-based contextual embeddings. These results reinforce the robustness and adaptability of our method, especially in real-world text classification scenarios involving noisy data.

4.4 Topic classification and spam detection results

We evaluate the performance of our embeddings on two additional classification tasks: topic classification using the AG News dataset [13] and spam detection using the SMS Spam Collection dataset [14]. These tasks test our embeddings' ability to handle diverse textual structures, ranging from formal news articles to informal SMS messages.

Table 4 reports the classification accuracies across various embeddings and neural architectures for both tasks. Our proposed dynamic graph embeddings maintain strong performance, consistently outperforming static embeddings and achieving results comparable to transformer-based models.

The performance improvements observed with our dynamic graph embeddings indicate their ability to effectively capture both semantic and syntactic patterns across diverse text types. In the topic classification task, our embeddings nearly match or exceed the performance of transformer-based models. Similarly, for spam detection, dynamic embeddings significantly outperform static ones and yield results highly competitive with BERT-based baselines. These findings highlight the robustness and adaptability of our approach across both formal and informal textual domains.

4.5 Named entity recognition (NER) results

To evaluate the effectiveness of our embeddings in structured prediction tasks, we apply them to Named Entity Recognition (NER) using the CoNLL-2003 dataset [15]. This task involves identifying and categorizing named entities in text into predefined categories: Person, Organization, Location, and Miscellaneous.

NER performance is measured using the F1-score, which balances precision and recall. Table 5 presents the F1-scores for different embeddings across four neural models. Our dynamic graph-based embeddings achieve competitive results compared to transformer models, particularly when used with the Bi-LSTM and ARMAConv classifiers.

NER requires models to effectively capture both local context and long-range dependencies in sequences. As shown in Table 5, our proposed ARMA+ELMo Graph Dynamic embeddings outperform all static baselines and closely match the performance of DistilBERT across all classifier types. Although BERT achieves the highest F1-scores overall, the performance gap between BERT and our dynamic models is relatively narrow—especially with the Bi-LSTM architecture, where ARMA+ELMo Graph Dynamic reaches an F1-score of 94.2% compared to BERT's 94.8%.

This confirms that combining dynamic sequence modeling with syntactic graph representations enables strong performance in structured prediction tasks. Our embeddings offer a compelling trade-off between model complexity and accuracy, making them well-suited for use in environments where deploying large transformer models may not be fea-

Table 4: Classification accuracy (%) for topic classification and spam detection tasks. Bold indicates the best performance in each column.

Embedding	Topic Classification				Spam Detection			
	CNN	Bi-LSTM	CNN+Bi-LSTM	ARMAConv	CNN	Bi-LSTM	CNN+Bi-LSTM	ARMAConv
Word2Vec [1]	88.5	89.3	89.1	88.9	96.2	96.8	96.5	96.3
GloVe [2]	88.8	89.5	89.3	89.0	96.5	96.9	96.7	96.5
FastText [6]	89.7	90.2	89.9	89.8	97.0	97.4	97.2	97.0
BERT [4]	92.4	**93.6**	93.1	92.9	**98.4**	**98.9**	98.7	98.6
DistilBERT [17]	91.8	93.0	92.7	92.5	98.0	98.6	98.4	98.3
ELMo-Like Dynamic (Ours)	90.3	91.2	90.8	90.6	97.5	98.0	97.7	97.5
ARMA Graph Dynamic (Ours)	90.7	91.5	91.0	90.8	97.6	98.2	97.9	97.7
ARMA+ELMo Graph Dynamic (Ours)	**92.5**	93.4	**93.2**	**93.0**	98.3	98.8	**98.7**	**98.6**

Table 5: F1-score (%) for Named Entity Recognition on the CoNLL-2003 dataset. Bold indicates the best result in each column.

Embedding	CNN	Bi-LSTM	CNN+Bi-LSTM	ARMAConv
Word2Vec [1]	88.1	90.2	89.7	88.9
GloVe [2]	88.5	90.4	89.9	89.1
FastText [6]	89.2	91.0	90.4	90.0
BERT [4]	**93.6**	**94.8**	**94.4**	**94.2**
DistilBERT [17]	93.0	94.1	93.8	93.5
ELMo-Like Dynamic (Ours)	90.5	92.7	91.8	91.2
ARMA Graph Dynamic (Ours)	91.1	93.0	92.4	91.8
ARMA+ELMo Graph Dynamic (Ours)	92.2	94.2	93.9	93.4

sible.

4.6 Intent classification results

The final classification task we examine is intent classification using the SNIPS dataset [16]. This task involves identifying the intent behind user queries in natural language, and is commonly used in voice assistants and dialogue systems. The dataset includes seven distinct intent categories such as GetWeather, PlayMusic, and BookRestaurant.

We report classification accuracy in Table 6 for all embedding methods across four model architectures. Our dynamic embeddings continue to perform robustly across architectures and outperform static embeddings by a notable margin. ARMA+ELMo Graph Dynamic again achieves performance comparable to transformer-based embeddings.

Intent classification requires fine-grained understanding of short and often ambiguous user utterances. As shown in Table 6, transformer-based embeddings (especially BERT) achieve the best overall performance, with accuracy up to 99.0% using the Bi-LSTM classifier. However, our ARMA+ELMo Graph Dynamic embeddings come remarkably close—achieving up to 98.9%—despite being significantly more lightweight and modular.

These results reinforce the capability of our dynamic embeddings to generalize well across intent-oriented tasks, offering a strong balance between performance and efficiency. Their flexibility makes them attractive for use in production environments such as mobile voice assistants or embedded NLP systems, where full transformer models may be impractical.

5 Discussion

The experimental results across five diverse NLP tasks—sentiment analysis, disaster tweet classification, topic classification, spam detection, named entity recognition, and intent classification—demonstrate the robustness and effectiveness of our proposed dynamic graph-based word embedding framework.

According to the effectiveness of dynamic graph-based embeddings, our approach consistently outperformed traditional static embeddings (Word2Vec, GloVe, FastText) and achieved competitive results when compared to contextual embeddings like BERT and DistilBERT. Notably, the ARMA+ELMo Graph Dynamic variant frequently achieved top-tier performance across all tasks, validating the benefit of combining sequence-based and graph-based contextual modeling. This hybrid approach captures both syntactic dependencies and dynamic semantic shifts within context—something that static embeddings inherently lack.

According to the performance across diverse tasks, the method's performance held consistently across tasks with varying characteristics, from short, noisy inputs (e.g., tweets and SMS messages) to long-form structured content (e.g., news and NER data). This suggests that the proposed dynamic embeddings generalize well across domains and linguistic complexities.

In comparing with Transformer-based embeddings, while BERT-based models often achieved slightly higher scores, our ARMA+ELMo dynamic embeddings delivered nearly equivalent performance in many settings—with the added advantage of being lighter-weight and easier to integrate into traditional neural pipelines. This is especially

Table 6: Intent classification accuracy (%) on the SNIPS dataset using different embedding methods and classifiers. Bold indicates the best performance in each column.

Embedding	CNN	Bi-LSTM	CNN+Bi-LSTM	ARMAConv
Word2Vec [1]	94.6	96.2	95.5	95.1
GloVe [2]	94.9	96.4	95.7	95.4
FastText [6]	95.6	96.8	96.2	95.8
BERT [4]	**98.4**	**99.0**	**98.8**	**98.7**
DistilBERT [17]	98.1	98.7	98.5	98.3
ELMo-Like Dynamic (Ours)	96.7	97.5	97.2	97.0
ARMA Graph Dynamic (Ours)	97.0	97.8	97.5	97.3
ARMA+ELMo Graph Dynamic (Ours)	98.2	98.9	98.7	98.6

beneficial in latency-sensitive or resource-constrained applications.

Another key observation is the adaptability of our embeddings across various classifier architectures. Whether used with CNNs, Bi-LSTMs, or GNN-based ARMAConv models, the proposed embeddings led to improved or competitive performance, confirming their architectural flexibility.

Future work could focus on optimizing model compression and exploring multilingual and cross-lingual extensions. Additionally, integrating our embeddings into generative or retrieval-augmented frameworks could be a promising direction. In summary, our dynamic graph-based word embedding framework presents a powerful and generalizable alternative to both static and large contextual models, offering a strong trade-off between performance, interpretability, and model size.

6 Conclusion

In this work, we presented an extended study of a dynamic graph-based word embedding framework initially proposed in our previous SOICT 2024 publication. The method combines static embeddings with dynamic features derived from next-word prediction models and integrates syntactic structure through undirected graph representations. Three embedding variants—ELMo-Like Dynamic, ARMA Graph Dynamic, and ARMA+ELMo Graph Dynamic—were introduced and evaluated extensively.

To validate the generalizability and effectiveness of our approach, we conducted comprehensive experiments across a wide range of NLP tasks, including sentiment analysis, disaster tweet classification, topic classification, spam detection, named entity recognition, and intent classification. The results consistently demonstrated that our dynamic embeddings outperform static baselines and are competitive with state-of-the-art transformer-based models such as BERT and DistilBERT.

Notably, our ARMA+ELMo Graph Dynamic embeddings achieved a classification accuracy of **93.2%** on the AG News topic classification task and an F1-score of **94.2%** on the CoNLL-2003 NER benchmark—results that are on par with, and in some cases surpass, those of larger pretrained models. These strong performances demonstrate the power of combining sequential and graph-based contextualization for semantic representation.

Our framework shows promising potential for applications in environments where model interpretability, training efficiency, and adaptability across tasks are critical. It serves as a scalable and flexible alternative to large-scale pretrained language models, especially in resource-constrained settings.

In future work, we plan to explore multilingual and cross-lingual extensions, investigate model compression techniques, and integrate our embedding strategy into large-scale retrieval-augmented and generative frameworks.

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013. [Online]. Available: https://doi.org/10.48550/arXiv.1301.3781

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: https://doi.org/10.3115/v1/d14-1162

[3] J. Sarzynska-Wawer, A. Wawer, A. Pawlak, J. Szymanowska, I. Stefaniak, M. Jarkiewicz, and L. Okruszek, "Detecting formal thought disorder by deep contextualized word representations," *Psychiatry research*, vol. 304, p. 114135, 2021. [Online]. Available: https://doi.org/10.1016/j.psychres.2021.114135

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/N19-1423

[5] R. E. Himes, H.-A. Tran, and T. X. Tran, "Leveraging dynamic graph word embedding for efficient con-

textual representations," in *International Symposium on Information and Communication Technology*. Springer, 2024, pp. 243–254. [Online]. Available: https://doi.org/10.1007/978-981-96-4288-5_20

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. [Online]. Available: https://doi.org/10.1162/tacl_a_00051

[7] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2)*, 2014, pp. 302–308. [Online]. Available: https://doi.org/10.3115/v1/p14-2050

[8] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 313–11 320. [Online]. Available: https://doi.org/10.1109/cvpr.2019.01157

[9] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional arma filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3496–3507, 2021. [Online]. Available: https://doi.org/10.1109/TPAMI.2021.3054830

[10] D. Grattarola and C. Alippi, "Graph neural networks in tensorflow and keras with spektral," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 99–106, 2021. [Online]. Available: https://doi.org/10.1109/MCI.2020.3039072

[11] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen, "Carer: Contextualized affect representations for emotion recognition," in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018, pp. 3687–3697. [Online]. Available: https://doi.org/10.18653/v1/D18-1404

[12] A. Howard, d. de Vries, P. Culliton, and Y. Guo, "Natural language processing with disaster tweets," Kaggle competition, 2019, https://www.kaggle.com/competitions/nlp-getting-started.

[13] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*, 2015, pp. 649–657.

[14] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of sms spam filtering: New collection and results," in *Proceedings of the 11th ACM Symposium on Document Engineering (DocEng '11)*, 2011, pp. 259–262. [Online]. Available: https://doi.org/10.1145/2034691.2034742

[15] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003, pp. 142–147. [Online]. Available: https://doi.org/10.48550/arXiv.cs/0306050

[16] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, T. Lavril, M. Primet, and J. Dureau, "Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1805.10190

[17] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019. [Online]. Available: https://arxiv.org/abs/1910.01108