An Encoder-Flexible Semantic Hypergraph Framework for API Recommendation in Mashup Development

Abhinav Jamwal[a], Dipender Singh[b], Manish Agrawal[c] and Sandeep Kumar[d]
Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee, India

Keywords: API Recommendation, Mashup Development, Semantic Recommendation, Sentence Encoders, Semantic Initialization, Web APIs.

Abstract: API recommendation is an important task in mashup development, where developers must identify suitable Web APIs from a large and evolving ecosystem. Existing methods mainly rely on invocation history, collaborative signals, or structural relations, but often underuse the semantic information available in mashup descriptions, API descriptions, and tag metadata. Moreover, many pipelines are tied to a single semantic encoder, which limits extensibility and obscures the effect of semantic backbone choice on recommendation quality. To address these limitations, this paper proposes an encoder-flexible semantic hypergraph framework for mashup–API recommendation. The framework encodes mashup descriptions, API descriptions, mashup tags, and API tags into dense semantic vectors, uses them to initialize trainable node representations, and refines them through multi-channel hypergraph propagation and adaptive layer aggregation. A joint objective further combines recommendation ranking with tag-level semantic alignment. Experiments with seven pretrained sentence encoders under the same downstream architecture show that the proposed framework is consistently effective and that encoder choice materially influences recommendation behavior. In particular, BGE and GTE yield the strongest ranking performance, whereas MPNet-based variants provide broader recommendation coverage. These findings highlight encoder-flexible semantic initialization as a meaningful design dimension in mashup API recommendation.

1 INTRODUCTION

Mashup development has become an important software construction paradigm in which developers build applications by integrating reusable Web APIs that provide complementary functionalities, such as social media access, mapping, payment support, weather information, and multimedia services (Alhosaini et al., 2024; Gao et al., 2015). In such settings, developers must identify suitable APIs from a large and evolving ecosystem where many services expose overlapping functionalities but differ in compatibility, quality, and contextual suitability (Xu et al., 2023; Gong et al., 2022). As a result, API selection during mashup construction is a non-trivial and time-consuming task, which has motivated substantial research on intelligent API recommendation techniques (Xia et al., 2014; Lian and Tang, 2022).

Existing API recommendation approaches have primarily relied on historical invocation patterns, collaborative filtering signals, co-occurrence statistics, or graph-based structural dependencies between mashups and APIs (Qi et al., 2021; Xiao et al., 2023; Kang et al., 2024). Although these methods capture useful evidence about functional compatibility and reuse behavior, they may not fully model the semantic relationship between mashup requirements and the capabilities of candidate APIs, especially when invocation histories are sparse, incomplete, or noisy (Alhosaini et al., 2024; Xu et al., 2023). This limitation is particularly important in practical mashup development, where developers often express intended functionality through short textual descriptions, while APIs are also characterized by concise descriptions and tags that reflect their functional scope (Qi et al., 2021; Wu et al., 2021).

To address this limitation, semantic information should be treated as a first-class source of evidence in API recommendation. Mashup descriptions can reflect developer intent, API descriptions can char-

[a] https://orcid.org/0000-0002-0213-3590
[b] https://orcid.org/0009-0009-5352-5453
[c] https://orcid.org/0009-0009-0216-2592
[d] https://orcid.org/0000-0002-3250-4866

Jamwal, A., Singh, D., Agrawal, M. and Kumar, S.
An Encoder-Flexible Semantic Hypergraph Framework for API Recommendation in Mashup Development.
DOI: 10.5220/0015044600004015

In *Proceedings of the 21st International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2026) - Volume 1*, pages 258-269
ISBN: 978-989-758-828-0; ISSN: 2184-4895

acterize functional capability, and tag metadata can provide compact cues about usage context and service semantics (Wang et al., 2023). When these heterogeneous textual artifacts are transformed into dense semantic representations, they can complement structural interaction patterns and improve recommendation quality, particularly when invocation history alone is insufficient (Zhang et al., 2024; Xiao et al., 2023). However, simply appending text embeddings as auxiliary features is often not enough. A more effective design is to integrate semantic representations directly into the representation learning process so that they can be refined jointly with relational information during training (Wang et al., 2024).

Recent progress in pretrained sentence encoders has created new opportunities for modeling short and heterogeneous software-related textual artifacts more effectively (Reimers and Gurevych, 2019; Song et al., 2020; Wang et al., 2020; Wang et al., 2022; Chen et al., 2024; Li et al., 2023). However, many recommendation pipelines remain tied to a single semantic backbone or treat semantic features as fixed inputs, which limits extensibility and makes it difficult to study how semantic initialization influences downstream recommendation quality. For mashup API recommendation, this is an important gap because the available textual artifacts are short, heterogeneous, and semantically compact, making encoder selection a practically meaningful design factor rather than a minor preprocessing choice.

In this paper, we propose an *encoder-flexible semantic hypergraph framework* for API recommendation in mashup development. The proposed framework first transforms mashup descriptions, API descriptions, mashup tags, and API tags into dense semantic vectors using a pretrained sentence encoder. These vectors are then used to initialize trainable node representations for mashups, APIs, mashup tags, and API tags. To capture higher-order structural dependencies, the framework constructs multiple similarity-induced hypergraph channels over mashup, API, mashup-tag, and API-tag relations, and refines the initialized representations through multi-layer hypergraph propagation and adaptive layer aggregation. The final model is jointly optimized to preserve recommendation-oriented compatibility while reinforcing tag-level semantic alignment.

A key characteristic of the proposed framework is its encoder-flexible design. Rather than coupling the recommendation architecture to a single semantic encoder, the framework separates semantic initialization from downstream hypergraph propagation and optimization, allowing alternative pretrained sentence encoders to be incorporated without modifying the overall recommendation pipeline. This design improves extensibility and reproducibility while also providing a principled foundation for studying how different semantic backbones shape recommendation behavior within a common architecture. Thus, encoder flexibility is not merely an implementation convenience, but an integral design property of the proposed framework.

The main contributions of this work are as follows:

- We propose an encoder-flexible semantic hypergraph framework for API recommendation in mashup development that unifies multi-source textual semantics and higher-order relational learning within a single recommendation architecture.
- We design a multi-source semantic initialization mechanism that encodes mashup descriptions, API descriptions, mashup tags, and API tags into trainable representations for downstream recommendation.
- We integrate semantic initialization with multi-channel hypergraph propagation, adaptive layer aggregation, and joint optimization to capture both structural compatibility and semantic alignment.
- We demonstrate that the proposed framework can be instantiated with alternative pretrained sentence encoders under a shared recommendation architecture, thereby enabling a systematic study of semantic backbone choice in mashup API recommendation.

The remainder of this paper is organized as follows. Section 2 describes the related work .Section 3 presents the proposed framework in detail. Section 4 describes the experimental setup and results. Section 5 discusses threats to validity. Section 6 provides further discussion of the findings. Finally, Section 7 concludes the paper.

2 RELATED WORK

This section reviews prior studies most relevant to the proposed framework from three perspectives. First, we discuss API recommendation in mashup development, which forms the core application setting of this work. Second, we review semantic and graph-based recommendation models that combine textual and structural information for representation learning. Third, we summarize recent advances in pretrained sentence encoders, which motivate the encoder-flexible design of the proposed framework.

2.1 API Recommendation in Mashup Development

API recommendation has been widely studied as a way to assist developers during mashup construction, where the goal is to identify suitable Web APIs that satisfy functional requirements while fitting the existing development context. Early studies explored category-based and service-oriented recommendation strategies, showing that historical mashup–API associations can provide useful signals for API discovery and selection (Xia et al., 2014; Gao et al., 2015). Later work further emphasized the practical difficulty of API recommendation in large and dynamic ecosystems, where many services provide overlapping functionality and manual selection becomes inefficient and error-prone (Alhosaini et al., 2024; Xu et al., 2023; Gong et al., 2022).

A major line of work has focused on recommendation methods based on invocation history, collaborative filtering, co-occurrence statistics, and compatibility-aware ranking. These methods learn recommendation patterns from observed mashup–API usage records and have shown promising performance in practice (Qi et al., 2021; Lian and Tang, 2022). However, their effectiveness may degrade when invocation data is sparse or when mashups and APIs have limited historical overlap (Alhosaini et al., 2024; Xu et al., 2023). This limitation motivates the need to incorporate richer semantic and contextual evidence into mashup API recommendation.

2.2 Semantic and Graph-Based Recommendation Models

To improve recommendation beyond simple interaction statistics, recent studies have incorporated textual semantics, graph structure, and higher-order relations into the representation learning process. Text-aware approaches exploit mashup descriptions, API descriptions, or tag metadata to better capture semantic compatibility between developer requirements and API functionality (Qi et al., 2021; Wu et al., 2021; Wang et al., 2023). These studies show that semantic information can complement historical invocation patterns, especially when structural evidence is incomplete or sparse.

In parallel, graph-based and hypergraph-based models have emerged as effective tools for capturing complex dependency patterns among mashups, APIs, and related entities. Graph neural approaches model structural relations through message passing over mashup–API interaction graphs, whereas hypergraph-based methods extend this idea to higher-order associations among multiple entities (Xiao et al., 2023; Kang et al., 2024; Zhang et al., 2024). More recent studies have attempted to combine semantic features with graph-based representation learning so that textual and structural signals can be optimized jointly rather than treated independently (Wang et al., 2024). However, many existing methods still use textual semantics as fixed auxiliary inputs or remain tied to a single semantic backbone, leaving limited room for modular semantic initialization and systematic comparison of encoder choices.

2.3 Pretrained Text Encoders for Semantic Representation

The growing success of pretrained sentence encoders has significantly improved the ability to represent short and heterogeneous textual artifacts in dense semantic spaces. Sentence-BERT established an efficient framework for learning semantically meaningful sentence embeddings suitable for similarity-based downstream tasks (Reimers and Gurevych, 2019). Subsequent encoder families, such as MPNet and MiniLM, improved contextual modeling and representation efficiency, making them attractive for semantic matching tasks under different computational budgets (Song et al., 2020; Wang et al., 2020). More recent embedding models, including E5, BGE, and GTE, have demonstrated strong semantic retrieval performance across a range of text representation settings (Wang et al., 2022; Chen et al., 2024; Li et al., 2023).

Despite these advances, encoder selection has not been sufficiently studied in mashup API recommendation. Existing recommendation pipelines often adopt a single pretrained encoder as a fixed design choice, making it difficult to compare semantic backbones under a shared downstream learning framework. In contrast, the proposed work treats encoder choice as an explicit architectural design dimension and adopts an encoder-flexible semantic hypergraph framework. This allows the same recommendation pipeline to be instantiated with alternative pretrained encoders while preserving the downstream propagation and optimization process.

Overall, prior studies have demonstrated the usefulness of interaction-driven recommendation, text-aware semantic modeling, and graph-based or hypergraph-based representation learning for API recommendation. However, semantic information is often incorporated in a limited or encoder-specific manner, and relatively few studies combine trainable semantic initialization, higher-order hypergraph propagation, and encoder flexibility within a unified API

Algorithm 1: Encoder-Flexible Semantic Hypergraph Framework for Mashup–API Recommendation.

Input: Mashups $\mathcal{M}$; APIs $\mathcal{A}$; mashup descriptions $\{D_m\}$; API descriptions $\{D_a\}$; mashup tags $\{T_m^M\}$; API tags $\{T_a^A\}$; training interactions $\mathbf{R}_{\text{train}}$; sentence encoder $\phi(\cdot)$; propagation depth L; thresholds $\alpha_1, \alpha_2, \alpha_3, \alpha_4$; epochs E; Top-N.

Output: Top-N API recommendation list $\pi(m)$ for each mashup m.

Encode mashup descriptions, API descriptions, mashup tags, and API tags using $\phi(\cdot)$, and initialize trainable embeddings $\mathbf{E}_m^{(0)}$, $\mathbf{E}_a^{(0)}$, $\mathbf{E}_{tm}^{(0)}$, and $\mathbf{E}_{ta}^{(0)}$. *//Semantic initialization*

Construct mashup-side, API-side, mashup-tag, and API-tag incidence matrices from $\mathbf{R}_{\text{train}}$, $\{T_m^M\}$, and $\{T_a^A\}$. Compute Jaccard similarities, threshold them using $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, and build normalized hypergraph propagation operators $\mathbf{P}_m$, $\mathbf{P}_a$, $\mathbf{P}_{tm}$, and $\mathbf{P}_{ta}$. *//Four hypergraph channels*

for $e = 1$ **to** E **do**

  Sample mashup–API triplets (m, a^+, a^-) and auxiliary positive/negative tag pairs.

  Set layer-0 embeddings from semantic initialization.

  for $\ell = 0$ **to** $L-1$ **do**

    Propagate embeddings over the four hypergraph channels.

  Aggregate layer-wise embeddings using learnable softmax weights. *//Adaptive layer aggregation*

  Compute mashup–API scores and optimize the joint objective

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{mt}} + \mathcal{L}_{\text{at}} + \lambda \|\Theta\|_2^2.$$

  Update model parameters by backpropagation.

Compute final mashup and API representations. For each target mashup m, rank all candidate APIs by inner-product score and return the Top-N list $\pi(m)$. *//Inference*

recommendation architecture. Motivated by these limitations, this work proposes an encoder-flexible semantic hypergraph framework that integrates multi-source semantic initialization, multi-channel hypergraph propagation, and joint optimization for recommendation and semantic alignment.

3 PROPOSED METHOD

This section presents the proposed encoder-flexible semantic hypergraph framework for API recommendation in mashup development. Algorithm 1 summarizes the overall procedure of semantic initialization, hypergraph construction, multi-layer propagation, joint optimization, and inference, while Figure 1 illustrates the corresponding architecture of the proposed framework.

At a high level, the proposed framework consists of five stages. First, mashup descriptions, API descriptions, mashup tags, and API tags are transformed into dense semantic representations using a pretrained sentence encoder. Second, these semantic vectors are used to initialize trainable node embeddings for mashups, APIs, mashup tags, and API tags. Third, similarity-induced hypergraph structures are constructed to model higher-order relations on the mashup side, API side, mashup-tag side, and API-tag side. Fourth, the initialized embeddings are refined through multi-layer hypergraph propagation and adaptive layer aggregation. Finally, the model is trained using a joint objective that combines recommendation ranking with tag-level semantic alignment.

3.1 Problem Definition and Notation

Let $\mathcal{M} = \{m_1, m_2, \ldots, m_{|\mathcal{M}|}\}$ denote the set of mashups and let $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ denote the set of Web APIs. The observed invocation relationships between mashups and APIs are represented by a binary interaction matrix

$$\mathbf{R} \in \{0,1\}^{|\mathcal{M}| \times |\mathcal{A}|},$$

where $R_{ij} = 1$ indicates that mashup m_i invokes API a_j, and $R_{ij} = 0$ otherwise.

Each mashup is further associated with a textual description and a set of category tags, while each API is associated with a textual description and a set of functional tags. Let $\mathcal{T}_m$ and $\mathcal{T}_a$ denote the mashup-tag and API-tag vocabularies, respectively. The goal of API recommendation is, for a target mashup m_i, to rank candidate APIs in $\mathcal{A}$ according to their predicted functional relevance and compatibility with m_i.

The proposed framework models four types of entities:

$$\mathcal{V} = \mathcal{M} \cup \mathcal{A} \cup \mathcal{T}_m \cup \mathcal{T}_a,$$

and learns dense representations for these entities by combining semantic initialization and hypergraph-based propagation.

3.2 Multi-Source Semantic Initialization

A central idea of the proposed framework is to treat textual semantics as a primary source of evidence for API recommendation. To this end, we exploit four

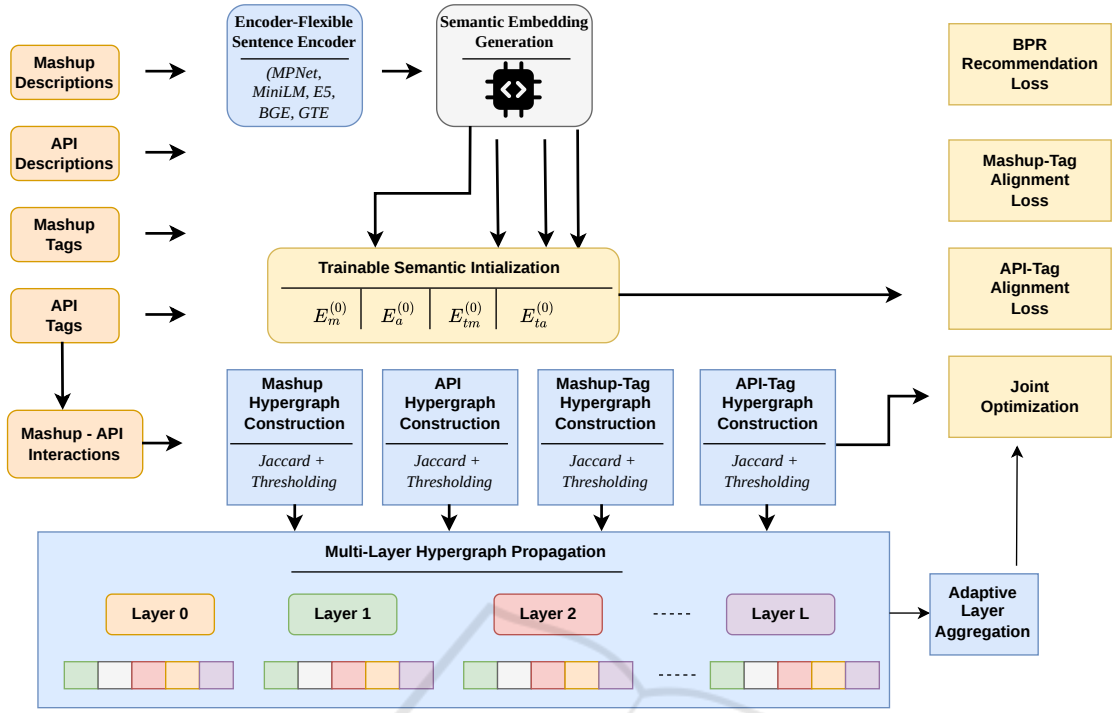

Figure 1: Architecture of the proposed encoder-flexible semantic hypergraph framework. Mashup descriptions, API descriptions, mashup tags, and API tags are encoded into trainable semantic initializations, refined through four similarity-induced hypergraph channels and multi-layer propagation, and aggregated adaptively to optimize recommendation and tag-alignment objectives for Top-N API recommendation.

textual channels, namely mashup descriptions, API descriptions, mashup tags, and API tags.

Let $\phi(\cdot)$ denote a pretrained sentence encoder. Given a textual input x, the encoder produces a dense semantic vector

$$\mathbf{z} = \phi(x) \in \mathbb{R}^d,$$

where d is the embedding dimension determined by the selected encoder backbone.

Using this encoder, we generate semantic vectors for each mashup description, each API description, each mashup tag, and each API tag. These vectors are then used to initialize four trainable embedding matrices:

$$\mathbf{E}_m^{(0)} \in \mathbb{R}^{|\mathcal{M}|\times d}, \quad \mathbf{E}_a^{(0)} \in \mathbb{R}^{|\mathcal{A}|\times d},$$
$$\mathbf{E}_{tm}^{(0)} \in \mathbb{R}^{|\mathcal{T}_m|\times d}, \quad \mathbf{E}_{ta}^{(0)} \in \mathbb{R}^{|\mathcal{T}_a|\times d}.$$

Here, $\mathbf{E}_m^{(0)}$ and $\mathbf{E}_a^{(0)}$ are initialized from mashup and API descriptions, respectively, whereas $\mathbf{E}_{tm}^{(0)}$ and $\mathbf{E}_{ta}^{(0)}$ are initialized from the textual content of mashup tags and API tags. Unlike static side features, these embeddings remain trainable during optimization, allowing the framework to refine semantic priors in a recommendation-aware manner. In this way, semantic information is embedded directly into the representation learning process rather than appended as external auxiliary input.

3.3 Similarity-Induced Hypergraph Construction

To model higher-order structural dependencies, the proposed framework constructs four similarity-induced hypergraph channels. These channels are derived from mashup–API interactions, mashup–tag associations, and API–tag associations.

3.3.1 Interaction and Incidence Matrices

From the observed interaction matrix $\mathbf{R}$, we derive

- a mashup-side interaction matrix $\mathbf{X}_m \in \{0,1\}^{|\mathcal{M}|\times|\mathcal{A}|}$, and
- an API-side interaction matrix $\mathbf{X}_a \in \{0,1\}^{|\mathcal{A}|\times|\mathcal{M}|}$.

Similarly, from mashup-tag and API-tag associations, we derive

- a mashup-tag incidence matrix $\mathbf{X}_{tm} \in \{0,1\}^{|\mathcal{T}_m|\times|\mathcal{M}|}$, and
- an API-tag incidence matrix $\mathbf{X}_{ta} \in \{0,1\}^{|\mathcal{T}_a|\times|\mathcal{A}|}$.

These four matrices define the structural basis for the mashup, API, mashup-tag, and API-tag channels, respectively.

3.3.2 Jaccard-Based Similarity Computation

For each channel, pairwise similarity is computed using the Jaccard coefficient. Given two binary incidence vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, their similarity is defined as

$$\text{Jaccard}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\|_1 + \|\mathbf{x}_j\|_1 - \mathbf{x}_i^\top \mathbf{x}_j}.$$

Using this formulation, we compute

- mashup–mashup similarity from $\mathbf{X}_m$,
- API–API similarity from $\mathbf{X}_a$,
- mashup-tag–mashup-tag similarity from $\mathbf{X}_{tm}$, and
- API-tag–API-tag similarity from $\mathbf{X}_{ta}$.

Each similarity matrix is then thresholded using the channel-specific thresholds α_1, α_2, α_3, and α_4 in order to retain only meaningful higher-order relations. The resulting thresholded similarities are used to form weighted incidence structures for the four hypergraph channels.

3.3.3 Normalized Hypergraph Operators

For a given channel, let $\mathbf{H}$ denote the corresponding weighted incidence matrix. The node-degree and hyperedge-degree matrices are defined as

$$\mathbf{D}_v(i,i) = \sum_e H(i,e), \qquad \mathbf{D}_e(e,e) = \sum_v H(v,e),$$

where $\mathbf{D}_v$ and $\mathbf{D}_e$ are diagonal degree matrices. The normalized propagation operator is then given by

$$\mathbf{P} = \mathbf{D}_v^{-1}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top.$$

In the proposed framework, separate operators are constructed for the four channels:

$$\mathbf{P}_m, \quad \mathbf{P}_a, \quad \mathbf{P}_{tm}, \quad \mathbf{P}_{ta},$$

corresponding to mashups, APIs, mashup tags, and API tags, respectively. Modeling these channels separately allows the framework to preserve distinct semantic and structural roles for each entity type.

3.4 Multi-Channel Hypergraph Propagation

After semantic initialization and hypergraph construction, the framework refines each entity representation through multi-layer propagation over its corresponding channel.

Let $\mathbf{E}^{(0)}$ denote the initial embedding matrix for a given entity type. At propagation layer l, the representation update is defined as

$$\mathbf{E}^{(l+1)} = \mathbf{P}\mathbf{E}^{(l)},$$

where $\mathbf{P}$ is the normalized propagation operator for the corresponding channel.

Accordingly, for the four channels we compute

$$\mathbf{E}_m^{(l+1)} = \mathbf{P}_m\mathbf{E}_m^{(l)}, \qquad \mathbf{E}_a^{(l+1)} = \mathbf{P}_a\mathbf{E}_a^{(l)},$$
$$\mathbf{E}_{tm}^{(l+1)} = \mathbf{P}_{tm}\mathbf{E}_{tm}^{(l)}, \qquad \mathbf{E}_{ta}^{(l+1)} = \mathbf{P}_{ta}\mathbf{E}_{ta}^{(l)}.$$

This propagation process allows each representation to aggregate information from structurally related entities under its corresponding higher-order context. In contrast to pairwise graph propagation, the hypergraph formulation better captures co-occurrence structures among mashups, APIs, and tag entities.

3.5 Adaptive Layer Aggregation

Different propagation depths may contribute differently to representation quality. Shallow layers tend to preserve semantic priors, whereas deeper layers may capture richer structural context. Instead of using only the final layer or averaging all layers uniformly, the proposed framework learns channel-specific aggregation weights.

For a given channel, let the layer-wise embeddings be

$$\mathbf{E}^{(0)}, \mathbf{E}^{(1)}, \ldots, \mathbf{E}^{(L)}.$$

These embeddings are aggregated as

$$\widetilde{\mathbf{E}} = \sum_{l=0}^{L} \beta_l \mathbf{E}^{(l)},$$

where

$$\beta_l = \frac{\exp(w_l)}{\sum_{k=0}^{L} \exp(w_k)}.$$

Here, $\{w_l\}$ are trainable scalar parameters and $\{\beta_l\}$ are softmax-normalized importance coefficients. Separate aggregation weights are learned for mashups, APIs, mashup tags, and API tags. The resulting final representations are denoted as

$$\widetilde{\mathbf{E}}_m, \quad \widetilde{\mathbf{E}}_a, \quad \widetilde{\mathbf{E}}_{tm}, \quad \widetilde{\mathbf{E}}_{ta}.$$

3.6 Recommendation Scoring

Given the final mashup representation $\widetilde{\mathbf{e}}_{m_i}$ and API representation $\widetilde{\mathbf{e}}_{a_j}$, the recommendation score is computed through inner-product matching:

$$\hat{y}_{ij} = \widetilde{\mathbf{e}}_{m_i}^\top \widetilde{\mathbf{e}}_{a_j}.$$

A larger score indicates a stronger predicted compatibility between mashup m_i and API a_j. During inference, candidate APIs are ranked in descending order according to $\hat{y}_{ij}$.

3.7 Joint Optimization with Tag-Level Semantic Alignment

The proposed framework is trained using a joint objective that combines recommendation ranking with two auxiliary semantic alignment losses. These auxiliary objectives help preserve semantically meaningful alignment during training, rather than allowing semantic initialization to be dominated solely by interaction-based ranking optimization.

3.7.1 Mashup–API Ranking Loss

For recommendation learning, we adopt a pairwise Bayesian Personalized Ranking (BPR) objective. Given a mashup m_i, a positive API a_j^+, and a negative API a_j^-, the ranking loss is defined as

$$\mathcal{L}_{\text{rec}} = -\sum_{(i,j^+,j^-)} \log \sigma\left(\widetilde{\mathbf{e}}_{m_i}^\top \widetilde{\mathbf{e}}_{a_j^+} - \widetilde{\mathbf{e}}_{m_i}^\top \widetilde{\mathbf{e}}_{a_j^-}\right),$$

where $\sigma(\cdot)$ denotes the sigmoid function. This objective encourages observed mashup–API pairs to receive higher scores than unobserved pairs.

3.7.2 Mashup–Tag Alignment Loss

To preserve semantic consistency between mashups and their associated tags, we introduce a mashup-tag pairwise loss. For a mashup m_i, a positive mashup tag $t_{m,k}^+$, and a negative mashup tag $t_{m,k}^-$, the loss is defined as

$$\mathcal{L}_{\text{mt}} = -\sum_{(i,k^+,k^-)} \log \sigma\left(\widetilde{\mathbf{e}}_{m_i}^\top \widetilde{\mathbf{e}}_{t_{m,k}^+} - \widetilde{\mathbf{e}}_{m_i}^\top \widetilde{\mathbf{e}}_{t_{m,k}^-}\right).$$

This objective encourages mashup representations to remain close to semantically relevant mashup-tag representations.

3.7.3 API–Tag Alignment Loss

Similarly, for API representations, we define

$$\mathcal{L}_{\text{at}} = -\sum_{(j,k^+,k^-)} \log \sigma\left(\widetilde{\mathbf{e}}_{a_j}^\top \widetilde{\mathbf{e}}_{t_{a,k}^+} - \widetilde{\mathbf{e}}_{a_j}^\top \widetilde{\mathbf{e}}_{t_{a,k}^-}\right),$$

where $t_{a,k}^+$ and $t_{a,k}^-$ are positive and negative API tags for API a_j. This loss helps API representations preserve functional semantic coherence with their associated API-tag representations.

3.7.4 Regularized Objective

The overall training objective is defined as

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{mt}} + \mathcal{L}_{\text{at}} + \lambda\|\Theta\|_2^2,$$

where Θ denotes the set of trainable parameters and λ is the regularization coefficient. By jointly optimizing these three losses, the framework learns representations that are simultaneously recommendation-aware and semantically aligned.

3.8 Encoder-Flexible Design

A key characteristic of the proposed framework is that the semantic encoding module is decoupled from the downstream hypergraph propagation and recommendation optimization stages. Specifically, the encoder is used only to generate the initial dense representations

$$\mathbf{E}_m^{(0)},\ \mathbf{E}_a^{(0)},\ \mathbf{E}_{tm}^{(0)},\ \mathbf{E}_{ta}^{(0)},$$

while all subsequent structural learning and scoring operations remain unchanged.

As a result, the framework can naturally support alternative sentence encoders without modifying the recommendation architecture itself. This encoder-flexible design enables the same recommendation pipeline to be instantiated with different semantic backbones under a shared training and inference procedure. The empirical comparison of encoder choices is presented later in the experimental section.

3.9 Computational Perspective

The proposed framework separates semantic representation generation from downstream recommendation training. Semantic embeddings for mashups, APIs, and tags are computed once during preprocessing and reused during model training. The main training cost, therefore, arises from hypergraph propagation and pairwise optimization. Since propagation is performed independently on four sparse channels, the framework remains computationally manageable for moderately sized mashup recommendation datasets while still capturing higher-order relations and multi-source semantics.

In general, the proposed method combines semantic representation learning, higher-order hypergraph propagation, adaptive layer aggregation, and joint semantic alignment within a unified encoder-flexible recommendation architecture.

4 EXPERIMENTAL SETUP AND RESULTS

This section evaluates the proposed encoder-flexible semantic hypergraph framework for API recommendation in mashup development. The experimental

study is designed to examine how different pretrained sentence encoders affect recommendation behavior when integrated into the same semantic hypergraph architecture. To ensure that the observed differences are attributable to semantic initialization rather than architectural variation, all experiments use the same hypergraph construction procedure, propagation depth, joint optimization strategy, and evaluation protocol, while varying only the sentence encoder used to generate the initial semantic representations.

The evaluation is conducted in a controlled and systematic manner. We first describe the dataset, evaluation metrics, and implementation details, and then present a detailed analysis of the results across different encoder backbones.

4.1 Dataset

We conduct experiments on the mashup API recommendation dataset used in R2API (Wang et al., 2025), which is derived from real-world Web API usage records; detailed dataset statistics are reported in that study. The dataset contains mashups, APIs, observed mashup-API invocation relationships, and associated textual metadata, including mashup descriptions, API descriptions, mashup tags, and API tags. These textual and tag-based artifacts are used for semantic initialization in the proposed framework.

We adopt a 10-fold cross-validation protocol. In each fold, mashups are partitioned into disjoint training and test subsets. The training subset is used to construct mashup–API, mashup-tag, and API-tag structures, while the test subset is used for recommendation evaluation on held-out mashups. For each held-out mashup, recommendations are generated using its available textual and tag information under the learned framework. The same fold partitioning is used for all encoder variants to ensure a fair comparison.

4.2 Evaluation Metrics

Following common practice in recommendation evaluation, we assess performance using Precision, Recall, Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), and Coverage at different recommendation cutoffs $N \in \{3,5,10,20\}$.

Precision@N measures the proportion of relevant APIs among the top-N recommendations, while Recall@N measures the proportion of relevant APIs successfully retrieved within the top-N list. MAP@N evaluates ranking quality by rewarding methods that place relevant APIs earlier in the recommendation list. NDCG@N further emphasizes the quality of ranked results by accounting for the positions of relevant APIs in the returned list. Coverage@N measures the proportion of distinct APIs that appear in the recommendation outputs, thereby reflecting the breadth of the recommendation space explored by the model.

Together, these metrics provide a balanced view of recommendation effectiveness, ranking quality, and recommendation diversity.

4.3 Implementation Details

The proposed framework is instantiated with seven pretrained sentence encoders drawn from the BGE (Chen et al., 2024), E5 (Wang et al., 2022), GTE (Li et al., 2023), MiniLM (Wang et al., 2020), and MPNet/Sentence-BERT (Song et al., 2020; Reimers and Gurevych, 2019) families. Specifically, we use BAAI/bge-base-en-v1.5, intfloat/e5-base-v2, thenlper/gte-base, sentence-transformers/all-MiniLM-L6-v2, sentence-transformers/all-MiniLM-L12-v2, sentence-transformers/all-mpnet-base-v2, and sentence-transformers/paraphrase-mpnet-base-v2.

For each encoder, semantic vectors are generated for mashup descriptions, API descriptions, mashup tags, and API tags. These vectors are then used to initialize the corresponding trainable embedding tables in the proposed framework. To ensure a controlled comparison, all non-encoder components are kept fixed across experiments, including hypergraph construction, propagation depth, joint optimization, and fold partitioning. The propagation depth is set to 3, the learning rate is set to 0.002, the batch size is set to 512, and the regularization coefficient is set to 10^{-4}. The similarity thresholds for the four hypergraph channels are fixed at $\alpha_1 = 0.6$, $\alpha_2 = 0.6$, $\alpha_3 = 0.8$, and $\alpha_4 = 0.8$. The model is trained for 400 epochs in each fold.

The dimensionality of the trainable initialization matches the output dimension of the selected encoder. In particular, the MiniLM-based encoders use 384-dimensional semantic initialization, whereas the remaining encoders use 768-dimensional initialization. Final results are reported as averages over all 10 folds.

4.4 Result Analysis

Table 1 reports the average recommendation performance of the proposed encoder-flexible semantic hypergraph framework across all 10 folds using different pretrained sentence encoders. Since the downstream architecture remains unchanged in all cases, the reported differences provide a direct view of how

Table 1: Average recommendation performance of the proposed encoder-flexible semantic hypergraph framework across 10 folds using different pretrained sentence encoders. The best result in each metric column within a Top-*N* block is shown in bold and the second-best result is underlined.

Top-N	Methods	P	R	MAP	NDCG	Cov
3	BGE-base	**0.3111**	**0.6736**	**0.7389**	**0.6755**	0.4372
	E5-base	0.3069	0.6626	0.7224	0.6610	0.3660
	GTE-base	0.3097	0.6705	0.7382	0.6746	0.4160
	MiniLM-L6	0.3033	0.6599	0.7227	0.6598	0.4298
	MiniLM-L12	0.3035	0.6591	0.7193	0.6585	0.4032
	MPNet-base	0.3025	0.6605	0.7227	0.6601	0.4766
	Para-MPNet	0.3028	0.6584	0.7174	0.6568	**0.4830**
5	BGE-base	0.2038	0.7157	**0.7385**	0.6893	0.5660
	E5-base	0.2023	0.7070	0.7216	0.6760	0.4926
	GTE-base	**0.2040**	**0.7185**	0.7382	**0.6906**	0.5904
	MiniLM-L6	0.1984	0.7011	0.7232	0.6734	0.5691
	MiniLM-L12	0.1990	0.7007	0.7198	0.6723	0.5777
	MPNet-base	0.1995	0.7046	0.7227	0.6755	**0.6457**
	Para-MPNet	0.2010	0.7076	0.7172	0.6742	0.6298
10	BGE-base	0.1125	0.7667	**0.7290**	0.7080	0.7830
	E5-base	0.1123	0.7609	0.7121	0.6959	0.7117
	GTE-base	**0.1132**	**0.7697**	0.7289	**0.7098**	0.7862
	MiniLM-L6	0.1104	0.7543	0.7141	0.6932	0.7777
	MiniLM-L12	0.1103	0.7510	0.7106	0.6910	0.7787
	MPNet-base	0.1108	0.7547	0.7137	0.6943	0.8298
	Para-MPNet	0.1114	0.7582	0.7083	0.6932	**0.8394**
20	BGE-base	**0.0614**	0.8119	0.7127	0.7223	0.9255
	E5-base	0.0609	0.8050	0.6983	0.7095	0.8894
	GTE-base	**0.0614**	**0.8138**	**0.7151**	**0.7236**	0.9245
	MiniLM-L6	0.0606	0.8002	0.6985	0.7079	0.9128
	MiniLM-L12	0.0603	0.7993	0.6970	0.7058	0.9170
	MPNet-base	0.0604	0.8028	0.7007	0.7090	**0.9500**
	Para-MPNet	0.0609	0.8062	0.6955	0.7081	0.9340

semantic initialization influences the behavior of the proposed framework. In other words, the purpose of this comparison is to evaluate how alternative semantic backbones affect the same recommendation architecture under a controlled setting.

Here, P, R, MAP, NDCG, and Cov denote Precision, Recall, Mean Average Precision, Normalized Discounted Cumulative Gain, and Coverage, respectively.

4.4.1 Encoder-Wise Performance Analysis

The results show that encoder selection has a consistent and measurable effect on recommendation quality. Among all evaluated backbones, `BAAI/bge-base-en-v1.5` and `thenlper/gte-base` achieve the strongest ranking-oriented performance across nearly all recommendation cutoffs. At the smallest and most practically important cutoff, `BAAI/bge-base-en-v1.5` yields the best Precision@3 (0.3111), Recall@3 (0.6736), MAP@3 (0.7389), and NDCG@3 (0.6755). This indicates that BGE produces the most accurate top-ranked API suggestions when only a very limited number of recommendations are shown to the developer.

At larger recommendation cutoffs, `thenlper/gte-base` becomes the strongest or nearly strongest encoder on several metrics. In particular, it achieves Precision@5 of 0.2040, Recall@5 of 0.7185, Recall@10 of 0.7697, Recall@20 of 0.8138, MAP@20 of 0.7151, and NDCG@20 of 0.7236. This suggests that GTE is especially effective at maintaining ranking quality as the recommendation list expands. Taken together, these observations indicate that BGE is particularly strong at optimizing the very top of the ranked list, whereas GTE provides the most balanced ranking behavior across short and long recommendation lists.

The results for `intfloat/e5-base-v2` also confirm that retrieval-oriented sentence encoders are well suited to the proposed framework. E5 performs competitively across all metrics, although it remains consistently below BGE and GTE. For example, it achieves Precision@3 of 0.3069, Recall@10 of 0.7609, and NDCG@20 of 0.7095. This shows that E5 is a robust semantic backbone within the framework, but its initialization appears to align slightly less effectively with downstream hypergraph propagation and joint ranking optimization than BGE and GTE.

4.4.2 Precision–recall–coverage Trade-Offs

A second important finding concerns the trade-off between ranking quality and recommendation coverage. The MPNet family, particularly `sentence-transformers/all-mpnet-base-v2`, does not achieve the best Precision, Recall, MAP, or NDCG values, but it consistently provides the strongest recommendation coverage at larger cutoffs. For instance, `all-mpnet-base-v2` attains Cov@5 of 0.6457, Cov@10 of 0.8298, and Cov@20 of 0.9500, which are the highest coverage values among all evaluated encoders. The same trend is also visible for `paraphrase-mpnet-base-v2`, which yields

Cov@10 of 0.8394 and Cov@20 of 0.9340.

This behavior is methodologically important because it reveals that semantic encoder selection affects not only the relevance of top-ranked items but also the breadth of the recommendation space explored by the model. In other words, some semantic backbones encourage the framework to focus more strongly on top-ranked accuracy, whereas others promote broader API exploration across the candidate space. From a practical perspective, this means that the proposed framework can support different recommendation goals depending on the selected encoder. If the primary objective is highly accurate top-ranked suggestions, BGE and GTE are preferable. If broader exploratory recommendation behavior is desired, MPNet-based encoders become more attractive.

4.4.3 Robustness across Encoder Families

Another important observation is the robustness of the proposed framework across diverse encoder families. Even the lighter MiniLM variants remain competitive despite their smaller representation size. For example, `all-MiniLM-L6-v2` achieves Recall@20 of 0.8002 and NDCG@20 of 0.7079, while `all-MiniLM-L12-v2` achieves Recall@20 of 0.7993 and NDCG@20 of 0.7058. Although these values do not exceed those of the strongest encoders, the performance gap remains moderate. This indicates that the proposed framework is not dependent on a single large semantic backbone and can preserve useful recommendation quality even with more compact encoders.

This robustness is a key methodological advantage of the proposed design. Because the framework separates semantic initialization from downstream structural learning, it remains stable across multiple encoder families while still preserving the ability to exploit semantic differences between them. This validates the central premise of the paper: encoder selection should be treated as an explicit design dimension in semantic API recommendation rather than as a fixed implementation detail.

5 THREATS TO VALIDITY

The findings of this study should be interpreted in light of several threats to validity. First, with respect to ***internal validity***, recommendation performance may be influenced by implementation choices and hyperparameter settings, such as similarity thresholds, propagation depth, learning rate, and regularization. To reduce this threat, all encoder variants are evaluated under the same architecture, optimization strategy, and fold partitioning, so that the observed differences can be attributed primarily to semantic initialization.

Second, regarding ***construct validity***, the study evaluates performance using Precision, Recall, MAP, NDCG, and Coverage. These metrics provide a balanced view of ranking quality and recommendation breadth, but they do not capture all practical aspects of API usefulness, such as compatibility constraints, documentation quality, or developer preferences. Therefore, the conclusions should be understood in terms of recommendation effectiveness under standard retrieval-oriented metrics.

Third, in terms of ***external validity***, the experiments are conducted on a single mashup API recommendation dataset derived from real-world Web API usage records. Although the dataset is appropriate for evaluating the proposed framework, the results may not generalize identically to other API ecosystems, domains, or recommendation settings with different sparsity patterns or richer contextual information.

Finally, with respect to ***conclusion validity***, the empirical study focuses on encoder-wise behavior within the proposed framework under a controlled setting. Thus, the conclusions are strongest for understanding how the choice of semantic backbone affects the same recommendation architecture, rather than for making universal claims about all API recommendation models or all sentence encoders.

6 DISCUSSION

The results provide three main insights into the proposed encoder-flexible semantic hypergraph framework. First, semantic initialization has a clear effect on recommendation performance even when the downstream hypergraph construction, propagation, and optimization settings remain unchanged. This shows that the semantic encoder is an important part of the overall recommendation architecture rather than merely a preprocessing component.

Second, different encoder families lead to different recommendation behaviors. BGE and GTE provide the strongest ranking-oriented performance, whereas MPNet-based variants consistently achieve higher coverage, and MiniLM variants remain competitive despite their lighter representation size.

Third, the results highlight the methodological value of the proposed framework. Because the architecture remains fixed while only the semantic backbone changes, the framework provides a clean setting for studying how semantic representations in-

teract with higher-order hypergraph learning. Overall, these findings show that the proposed approach is both effective and flexible, and that encoder-flexible semantic initialization is a meaningful design dimension in mashup API recommendation.

7 CONCLUSION AND FUTURE WORK

This paper presented an encoder-flexible semantic hypergraph framework for API recommendation in mashup development. The proposed framework integrates multi-source textual semantics with higher-order relational learning through semantic initialization, multi-channel hypergraph propagation, adaptive layer aggregation, and joint optimization. The results show that the framework is effective across different semantic backbones and that encoder choice has a clear impact on recommendation behavior. In particular, BGE and GTE achieve the strongest ranking-oriented performance, whereas MPNet-based variants provide broader recommendation coverage.

Future work will focus on extending the framework with additional semantic backbones and richer contextual signals, such as API evolution, temporal interaction patterns, and compatibility information. It would also be valuable to investigate the applicability of the proposed framework to other software engineering recommendation tasks.

REFERENCES

Alhosaini, H., Alharbi, S., Wang, X., and Xu, G. (2024). Api recommendation for mashup creation: A comprehensive survey. *The Computer Journal*, 67(5):1920–1940.

Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., and Liu, Z. (2024). Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 4(5).

Gao, W., Chen, L., Wu, J., and Gao, H. (2015). Manifold-learning based api recommendation for mashup creation. In *2015 ieee international conference on web services*, pages 432–439. IEEE.

Gong, W., Zhang, X., Chen, Y., He, Q., Beheshti, A., Xu, X., Yan, C., and Qi, L. (2022). Dawar: Diversity-aware web apis recommendation for mashup creation based on correlation graph. In *Proceedings of the 45th international ACM SIGIR conference on Research and Development in information retrieval*, pages 395–404.

Kang, G., Wang, Y., Ren, H., Cao, B., Liu, J., and Wen, Y. (2024). Ks-gnn: keyword search via graph neural network for web api recommendation. *IEEE Transactions on Network and Service Management*, 21(5):5464–5474.

Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., and Zhang, M. (2023). Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Lian, S. and Tang, M. (2022). Api recommendation for mashup creation based on neural graph collaborative filtering. *Connection science*, 34(1):124–138.

Qi, L., Song, H., Zhang, X., Srivastava, G., Xu, X., and Yu, S. (2021). Compatibility-aware web api recommendation for mashup creation via textual description mining. *ACM Transactions on Multimidia Computing Communications and Applications*, 17(1s):1–19.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2020). Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. (2022). Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

Wang, X., Xi, M., Li, Y., Pan, X., Wu, Y., Deng, S., and Yin, J. (2024). Sehgn: Semantic-enhanced heterogeneous graph network for web api recommendation. *IEEE Transactions on Services Computing*, 17(5):2836–2849.

Wang, Y., Chen, J., Huang, Q., Xia, X., and Jiang, B. (2023). Deep learning-based open api recommendation for mashup development. *Science China Information Sciences*, 66(7):172102.

Wang, Y., Kang, X., Li, X., and Gao, S. (2025). R2api: A novel method for web api recommendation by using hgnns with multi-task learning. *IEEE Transactions on Software Engineering*.

Wu, H., Duan, Y., Yue, K., and Zhang, L. (2021). Mashup-oriented web api recommendation via multi-model fusion and multi-task learning. *IEEE Transactions on Services Computing*, 15(6):3330–3343.

Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., and Wu, C. (2014). Category-aware api clustering and distributed recommendation for automatic mashup creation. *IEEE Transactions on Services Computing*, 8(5):674–687.

Xiao, G., Fei, J., Li, D., Wang, C., Cheng, Z., and Lu, J. (2023). Mrhn: Hypergraph convolutional network for web api recommendation. In *2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 179–184. IEEE.

Xu, G., Lian, S., and Tang, M. (2023). Web api service recommendation for mashup creation. *International Journal of Computational Science and Engineering*, 26(1):45–53.

Zhang, C., Qin, S., Wu, H., and Zhang, L. (2024). Cooperative mashup embedding leveraging knowledge graph for web api recommendation. *IEEE Access*, 12:49708–49719.