ITEM IDENTIFIABILITY FROM LARGE-LANGUAGE-MODELS-BASED EMBEDDINGS IN RECOMMENDER SYSTEMS

Bc. Linda Beková

Master's thesis
Faculty of Information Technology
Czech Technical University in Prague
Department of Applied Mathematics
Study program: Informatics
Specialisation: Knowledge Engineering
Supervisor: Dr. rer. nat. Rodrigo Augusto da Silva Alves
May 7, 2026

FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of master's thesis

Title:	Item Identifiability from Large-Language-Models-Based Embeddings in Recommender Systems
Student:	Bc. Linda Beková
Supervisor:	Dr. rer. nat. Rodrigo Augusto da Silva Alves
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2026/2027

Instructions

Recommender systems are a core component of modern online platforms, enabling personalized discovery across large item catalogs. In many machine learning–based recommender systems, side information is incorporated through dense embeddings, often generated by Large Language Models (LLMs), to represent items while abstracting away their original content. These embeddings are commonly treated as opaque representations that obscure the identity and descriptive information of items.

This project studies the identifiability of items from LLM-generated embeddings and investigates under which conditions item identity or content can be inferred. Several realistic scenarios will be considered, including cases where the set of items is known but item identifiers are not, where items are unknown, where the LLM used to generate embeddings is known or unknown, and where only partial text used to generate item embeddings is available. Analyzing these scenarios provides insights into the privacy, robustness, and interpretability of embedding-based recommender systems.

Proposed Tasks

1) Review existing literature on LLM-based item embeddings in recommender systems, with a focus on privacy, information leakage, and embedding inversion.

2) Formalize different item-identifiability scenarios based on the availability of item sets,

---

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 4 February 2026 in Prague.

identifiers, generating LLMs, and textual inputs.

3) Design and implement methods to infer item identity or content from embeddings under the defined scenarios (for at least two publicly available datasets).

4) Evaluate the proposed methods quantitatively and discuss the implications for privacy, security, and the practical deployment of embedding-based recommender systems.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 4 February 2026 in Prague.

I would like to sincerely thank my advisor, Dr. rer. nat. Rodrigo Augusto da Silva Alves for his guidance and valuable feedback throughout the preparation of this thesis. I am also deeply thankful to Patrik for his never-ending encouragement, understanding, and support. Lastly, I would like to express my gratitude to all close ones for their continuous support throughout my journey at FIT CTU over the past few years.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act. I declare that I have used AI tools during the preparation and writing of my thesis. I have verified the generated content. I confirm that I am aware that I am fully responsible for the content of the thesis.

In Prague on May 7, 2026

Abstract

Large language models are increasingly used in recommender systems to generate item embeddings from textual metadata. Although these embeddings are often treated as privacy-preserving, this thesis shows that they can still reveal information about the underlying items. We study item-level identifiability and evaluate whether an attacker can identify the embedding model and match leaked item embeddings to candidate items under different knowledge scenarios. Experiments on MovieLens and Goodreads Books datasets show that semantics can be revealed through embeddings, especially when titles or detailed descriptions are available. We provide and evaluate a defence that reduces the success of attacks.

Keywords recommender systems, large language models, item embeddings, embedding privacy, item identification, model identification, embedding leakage

Abstrakt

Velké jazykové modely se v doporučovacích systémech stále častěji používají ke generování embeddingů položek z textových metadat. Přestože jsou tyto embeddingy často považovány za dostatečné z hlediska ochrany soukromí, tato práce ukazuje, že stále mohou odhalovat informace o původních položkách. Zabýváme se identifikovatelností na úrovni položek a hodnotíme, zda útočník za různých předpokladů dokáže jednak identifikovat použitý embedding model, jednak odhalit původní položku. Experimenty na datasetech MovieLens a Goodreads Books ukazují, že sémantika položek může být odhalena skrze embedding, zejména pokud jsou k dispozici názvy nebo jejich podrobné popisy. Navrhujeme a vyhodnocujeme obranu, která snižuje úspěšnost útoků.

Klíčová slova doporučovací systémy, velké jazykové modely, embeddingy položek, soukromí embeddingů, identifikace položek, identifikace modelu, únik informací z embeddingů

Contents

List of Figures

List of Tables

List of abbreviations

LLM	Large Language Model
RecSys	Recommender Systems
NLP	Natural Language Processing
HR@1	Hit Rate at 1
MRR	Mean Reciprocal Rank
MSE	Mean Squared Error
RMSE	Root Mean Squared Error

Chapter 1

Introduction

Large Language Models (LLMs) have been widely adopted across a variety of domains, achieving state-of-the-art performance in tasks such as natural language understanding, generation, and information retrieval [1]. One domain that has increasingly benefited from these advances is Recommender Systems (RecSys), where LLMs are used to enhance representations of users and items, resulting in more accurate, context-aware recommendations [2]. However, alongside these benefits, integrating LLMs introduces new privacy and security vulnerabilities that require careful investigation.

Modern recommender systems extend beyond traditional user-item interaction matrices by incorporating rich textual features describing users and items [1]. To facilitate scalable deployment and, in some cases, to preserve data privacy, such information is often shared with third-party service providers as LLM-generated embeddings. These embeddings are commonly assumed to be privacy-preserving, as they are high-dimensional and not directly interpretable by humans [3].

However, this assumption has recently been challenged. Prior work has demonstrated that embeddings can, under certain conditions, be inverted to recover the original input text. These so-called embedding inversion attacks have been explored under various threat models, differing in the amount of auxiliary information available to the attacker. Existing approaches typically assume access to additional resources, such as the ability to query the embedding model or the availability of at least one known text-embedding pair. Under such assumptions, studies have shown that recovering semantically similar or even near-identical text is feasible, raising an important question: to what extent do these embeddings actually conceal the original information?

In this work, we aim to investigate a more constrained and arguably more realistic attack scenario focused specifically on tabular data from recommender systems. Rather than focusing on direct inversion of embeddings, we consider whether an attacker can exploit their structural properties in the context of RecSys. Specifically, we examine whether it is possible to identify the underly-

ing embedding model used to generate the representations and, subsequently, leverage this information to infer which items correspond to a given set of embeddings. By shifting the focus from exact text reconstruction to item-level identification, we offer a complementary perspective on the privacy risks of embedding-based data sharing. In particular, we raise the question: If an attacker gains access to the shared embeddings, can they recover or infer the underlying items? The main objectives and scientific contributions of this work are summarised as follows:

Proposed Tasks

1. **Review existing literature on LLM-based item embeddings in recommender systems, with a focus on privacy, information leakage, and embedding inversion.** – The fundamental information necessary for the thesis and related literature to embedding inversion attacks is explored in Chapter 2.
2. **Formalise different item-identifiability scenarios based on the availability of item sets, identifiers, generating LLMs, and textual inputs.** – The formal definition of selected scenarios with varying information available to the adversary is described in Chapter 3.
3. **Design and implement methods to infer item identity or content from embeddings under the defined scenarios (for at least two publicly available datasets)** – The implementation details, including the selected datasets, models, and experimental setup, are described in Chapter 4.
4. **Evaluate the proposed methods quantitatively and discuss the implications for privacy, security, and the practical deployment of embedding-based recommender systems.** – Results of explored approaches are further described in Chapter 4, followed by a discussion on what setups have an effect on the feasibility of these attacks.

Chapter 2

Fundamentals and Advances

This chapter introduces large language models and transformer-based embedding models, their role in recommender systems, and the associated privacy risks of textual embeddings.

2.1 Large Language Models

Natural language processing (NLP) provides the foundation for enabling machines to interpret and generate human language. In recent years, large language models have achieved state-of-the-art performance across a wide range of NLP tasks, driven by advances in transformer architectures, the availability of large-scale training data, and increased computational resources [1].

Modern language models are typically trained using self-supervised objectives, such as next-token prediction or masked language modelling[1], and can be adapted to downstream tasks through fine-tuning or prompting [5]. In prompting-based approaches, additional contextual information, such as task descriptions or example inputs, is provided at inference time to guide model behaviour [6, 3]. This flexibility has enabled LLMs to be applied across a variety of domains, including recommender systems.

Pre-trained language models can be broadly categorised based on their architecture into encoder-only models (e.g., BERT), decoder-only models (e.g., GPT), and encoder–decoder models (e.g., T5) [2]. Among these, encoder-based models are particularly relevant for representation learning [7], as they produce contextualised token embeddings that can be aggregated into fixed-dimensional vector representations. In contrast, decoder-only and encoder–decoder models are more commonly associated with generative tasks [8], although they can also be adapted for embedding generation [9].

[1] Masked language modelling is a pre-training objective in which some input tokens are randomly replaced by masks, and the model is trained to predict the original vocabulary IDs of those tokens using the surrounding context [4].

LLMs can be viewed as a natural extension of transformer-based language models, characterised primarily by their scale. Contemporary models often contain billions of parameters and are trained on massive corpora comprising diverse textual sources, such as Wikipedia, GitHub, and scientific publications. This increase in scale has led not only to improved performance on standard benchmarks but also to the emergence of new capabilities, including few-shot learning[2], instruction following, and advanced reasoning behaviours. These properties have motivated the integration of LLMs into downstream applications, including modern recommender systems [2].

2.2 Turning Text into Vectors

Embedding models play a central role in modern machine learning by mapping raw inputs into dense vector representations that capture semantic relationships. In NLP, these representations are designed such that semantically similar inputs are located close to each other in the embedding space [5]. This property underpins many applications, including search, clustering, and recommendation.

Two primary types of embedding models are commonly used:

Word embedding models. These operate at the level of individual tokens, assigning each word a vector representation in a continuous space. Depending on the model, these representations may be static or context-dependent.

Sentence embedding models. These operate on entire text sequences, producing fixed-dimensional representations of sentences or short documents. Typically, token-level representations are first computed and then aggregated – often via pooling – into a single vector $\phi(x) \in \mathbb{R}^d$.

A key distinction in embedding methods is between *static* and *contextual representations*. Static embeddings assign a single vector to each word, independent of context, which limits their ability to capture polysemy[3]. For example, the word "bat" would be represented identically regardless of whether it refers to an animal or sports equipment.

Contextual embedding models address this limitation by first mapping each token to an initial, context-independent embedding and then transforming these embeddings into contextualised hidden representations [10]. Formally, given a sequence of tokens $S = (t_1, \ldots, t_N)$, each token t_i is first mapped to an input embedding e_{t_i}. A contextual encoder then processes the full sequence of input embeddings $(e_{t_1}, \ldots, e_{t_N})$ and produces contextualised representations $(h_{t_1}, \ldots, h_{t_N})$:

$$(h_{t_1}, \ldots, h_{t_N}) = f(e_{t_1}, \ldots, e_{t_N}).$$

[2] Few-shot learning enables a model to perform a task by conditioning it on a small number of examples, without requiring task-specific retraining [6].

[3] A single word which can have different meanings depending on the context.

Thus, unlike static word embeddings, the final representation h_{t_i} of a token depends not only on the token itself, but also on the surrounding tokens in the input sequence. These contextualised representations, implemented by architectures such as transformers or recurrent neural networks, capture richer semantic and syntactic information, making them more effective for downstream tasks.

Prior to embedding, text is typically segmented into tokens using subword-based tokenisation methods such as SentencePiece [11]. Traditional embedding methods, such as Word2Vec [12] and GloVe [13], learn a global embedding matrix $E \in \mathbb{R}^{V \times d}$, where each row corresponds to a word in the vocabulary. In contrast, modern contextual models dynamically generate embeddings from the full input sequence, enabling more expressive representations.

Sentence embeddings can be derived from contextual token representations [5] by applying a pooling operation over the sequence:

$$z = \phi(x) = \text{pool}(h_{t_1}, \ldots, h_{t_n}).$$

Mean pooling is a commonly used approach and has been shown to produce effective sentence representations for semantic similarity tasks [7]. Since these embeddings are explicitly designed to preserve semantic information, they may also retain substantial information about the original input. A visual representation of how transformer-based language model encoders are used to create sentence embeddings is shown in Figure 2.1, which demonstrates a simplified version of the transformer architecture.

This property is particularly relevant to privacy. While embedding representations are often treated as anonymised or non-interpretable, their ability to encode rich semantic content makes them susceptible to information leakage. In particular, embedding inversion attacks exploit this property by attempting to reconstruct the original input from its vector representation. As a result, embeddings that are more informative for downstream tasks may also pose greater privacy risks.

In this thesis, the term *LLM-based embeddings* refers broadly to textual representations produced by transformer-based language model encoders. Although LLMs are often associated with generative architectures, encoder-based models are widely used to construct semantic embeddings for downstream tasks. In the experimental part of this work, item representations are generated using Hugging Face sentence transformer models [14], which map textual descriptions to fixed-dimensional embedding vectors.

2.3 Recommendation Essentials

Building on the representation-learning capabilities of transformer-based models discussed above, RecSys leverages these embeddings to model user preferences and item characteristics.

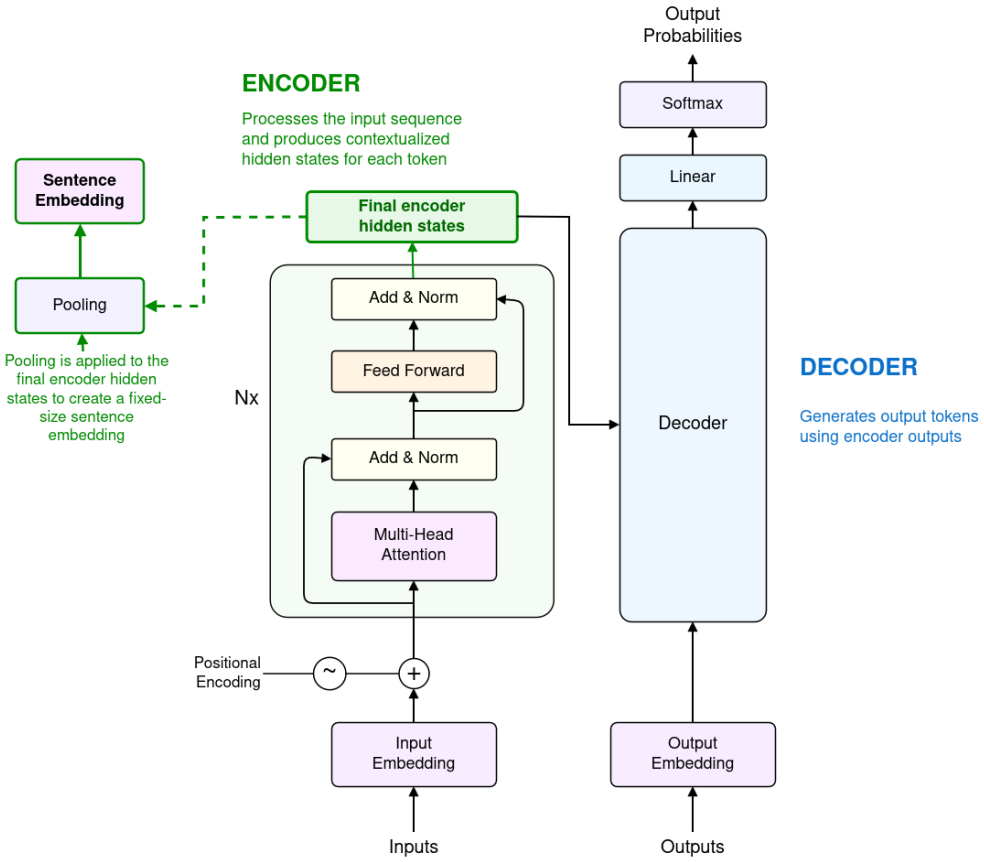

Figure 2.1 Sentence embedding creation from transformer hidden states

Recommender systems aim to model user preferences and suggest relevant items from a large catalogue. Formally, a recommender system is defined over a set of users $\mathcal{U} = \{u_1, \ldots, u_m\}$ and a set of items $\mathcal{I} = \{i_1, \ldots, i_n\}$, with interactions represented by a partially observed matrix $R \in \mathbb{R}^{m \times n}$. Each entry R_{ui} captures the observed interaction or preference of user u for item i, if available. In practice, this matrix is highly sparse, as users typically interact with only a small fraction of all items.

To address this sparsity, modern recommender systems often incorporate additional side information, such as user features $X \in \mathbb{R}^{m \times p}$ and item features $Y \in \mathbb{R}^{n \times q}$, which may include textual descriptions, metadata, or contextual signals.

The objective of a recommender system is to estimate preferences for unobserved user–item pairs and produce a ranked list of items [2]. Formally, for a user $u \in \mathcal{U}$, context c, and candidate set $\mathcal{I}$, the system can be expressed as

$$[i_k]_{k=1}^{N} \leftarrow RS(u, c, \mathcal{I}),$$

where RS denotes the recommendation function and $[i_k]_{k=1}^{N}$ is the ranked list of recommended items. The context c may include additional signals such as temporal, spatial, or session-based information. Items appearing earlier in the list are assumed to have higher predicted relevance.

While this representation abstracts the recommendation task, modern systems implement it through complex pipelines that integrate multiple data sources and modelling components.

2.4 Recommendation Pipeline

Modern recommender systems, particularly those based on deep learning, are typically structured as multi-stage pipelines composed of several interconnected components [2]. These components transform raw data into actionable recommendations:

Data Collection. User interactions (e.g., clicks, ratings, purchases) and auxiliary data (e.g., item attributes, user profiles, contextual signals) are collected. This stage defines the raw information available to the system.

Feature Engineering. Raw data is processed and transformed into structured representations. This includes cleaning, selecting, and augmenting features, as well as handling different modalities such as text, images, or categorical data.

Feature Encoding. Structured features are mapped into dense vector representations. This stage is particularly important, as it determines how information is represented and subsequently used by downstream models.

Scoring and Ranking. The system estimates the relevance of items for a given user and produces a ranked list. Methods such as collaborative filtering, sequential modelling, and neural ranking architectures are commonly used.

User Interaction. Recommendations are presented to users, and their responses are used to update the system.

Pipeline Control. The overall process is coordinated by a controller that manages interactions between components and optimises the recommendation workflow.

Among these stages, feature encoding plays a central role, as it directly determines how user and item information is represented. With the introduction of LLMs, this stage has undergone a significant transformation.

2.5 LLM-Powered Recommendation: Roles inside the Pipeline

The increasing availability of textual data in recommender systems, combined with advances in large language models, has led to their integration across multiple stages of the recommendation pipeline [2]. Therefore, the term LLM-based RecSys can have different meanings depending on where the LLM is applied in the recommendation pipeline. LLMs can be used for feature engineering, representation learning, scoring, and user interaction.

In this thesis, we focus primarily on the feature encoding stage, where textual information is transformed into dense semantic embeddings. This stage is particularly relevant, as it defines the representations later used for ranking and recommendation, thereby directly influencing both system performance and potential privacy risks.

LLMs for Feature Engineering and Data Augmentation LLMs can enhance feature engineering by generating additional information from raw inputs, such as item descriptions or user interaction histories. By leveraging their linguistic and world knowledge, they can improve representation quality and mitigate issues such as data sparsity.

In particular, LLMs can be used for *user and item-level feature augmentation*, where they generate descriptive attributes, extract semantic aspects, or summarise user preferences [15]. These augmented features can improve downstream modelling. Additionally, LLMs can support *instance-level sample generation* by producing synthetic data [16], such as simulated interactions or item descriptions, thereby enhancing robustness and dataset coverage.

While feature engineering improves the quality of input data, the system's effectiveness ultimately depends on how this information is encoded into representations.

LLMs as Feature Encoders Feature encoding is one of the most critical stages in modern recommender systems, as it determines how information is represented and shared across the pipeline.

LLMs enable the encoding of textual data into semantically rich embeddings, providing two key advantages: improved representation quality and the ability to operate in a shared semantic space across domains. This leads to *representation enhancement*, where textual information, such as item descriptions or user-generated content, is encoded into dense embeddings, thereby improving content understanding and recommendation quality [17]. These representations can also be extended to model user preferences based on interaction histories.

Furthermore, LLM-based encoders enable *cross-domain recommendation* by mapping heterogeneous data into a shared embedding space, facilitating transfer learning across domains [18]. This is particularly beneficial in cold-start[4] and long-tail scenarios[5].

However, the same properties that make embeddings effective – namely, their ability to capture rich semantic information – also make them a potential source of information leakage.

LLMs as Scoring and Ranking Functions LLMs can also be used to estimate relevance or directly generate recommendations. These approaches can be broadly categorised into scoring-based, generation-based, and hybrid methods.

In scoring-based approaches, LLMs are used for *item scoring*, where they predict relevance scores for user–item pairs using adapted architectures [21]. In generation-based approaches, LLMs perform *item generation*, directly producing recommendations [22], either in an open-set setting or by ranking a predefined candidate set. Hybrid approaches combine both paradigms, leveraging the multi-task capabilities of LLMs to support both scoring and generation [23].

LLMs for User Interaction LLMs enable interactive recommendations through dialogue, allowing systems to dynamically refine user preferences. In *task-oriented interaction*, LLMs interpret user intent and assist in decision-making [24], improving recommendation accuracy and explainability. In contrast, *open-ended interaction* settings allow LLMs to guide conversations and gradually infer user preferences over time [25].

[4]Cold-start refers to recommendation settings in which insufficient interaction data is available, such as when an item has not yet received any ratings, when a new user has no observed preference history, or when a system has little or no historical purchase or rating data [19].

[5]Long-tail scenarios refer to recommendation settings dominated by items with sparse interaction data, often because they are unpopular, niche, or newly introduced [20].

LLMs as Pipeline Controllers Recent work often explores using LLMs as pipeline controllers to coordinate multiple components of the recommender system. By leveraging capabilities such as reasoning and tool usage, LLMs can dynamically manage system behaviour, decide when to invoke specific modules, and refine outputs through filtering or re-ranking [26]. This enables more adaptive, flexible, and autonomous recommender systems.

Privacy of LLM-powered Recommender Systems The integration of LLMs into recommender systems introduces new privacy challenges. While embeddings are often treated as anonymised representations, they may retain substantial semantic information about item content or user preferences.

As a result, embeddings shared across system components or with external services may expose sensitive information. An adversary with access to such embeddings may be able to infer attributes, reconstruct textual content, or identify the corresponding items.

Existing mitigation strategies include differential privacy, access control, and secure computation. However, the extent to which transformer-based embeddings preserve recoverable information remains an open research question. This motivates the investigation in this thesis, which examines whether embeddings can be exploited to identify underlying items in recommender systems.

2.6 Prior Embedding Attacks

Embeddings are often regarded as less privacy-sensitive than raw text because they are high-dimensional, continuous, and not directly interpretable by humans [5]. This perception has motivated their use in settings where raw data is considered too sensitive to share directly. For example, embeddings may be computed locally on user devices and subsequently transmitted to service providers or third parties, thereby replacing the original input text with a learned vector representation. However, this transformation does not necessarily eliminate sensitive information. Since embeddings are explicitly designed to preserve the semantic properties of the input, they may retain information about the original text that an adversary can exploit. This raises two central questions: what information remains encoded in the embedding, and what auxiliary knowledge is required to extract it?

Song and Raghunathan [5] provide a taxonomy of attacks against embeddings that aim to infer private information from such representations. They distinguish between three main classes of attacks:

Sensitive attribute inference attacks. In this setting, the adversary aims to infer a sensitive attribute s^* associated with an unknown input text x^*, given only its embedding $\phi(x^*)$.

Membership inference attacks. The adversary seeks to determine whether a target data point x^* was included in the training data of the embedding

model. For word embedding models, membership at the level of individual words is often less meaningful, since such models are typically trained on large vocabularies or near-complete dictionaries. Song and Raghunathan [5] therefore formulate membership in terms of the contexts in which words appeared during training, rather than the mere presence of individual words.

Embedding inversion attacks. In this type of attack, the adversary attempts to reconstruct the original input text, or a semantically similar approximation, from a leaked or observed embedding. We discuss embedding inversion attacks in more detail in Section 2.6.1.

Among these attack classes, embedding inversion is particularly relevant to the privacy risks studied in this work, because it directly concerns the extent to which an embedding representation can reveal information about the input from which it was generated. While our work does not aim to reconstruct text word-for-word, embedding inversion provides the closest conceptual foundation for analysing how much information is preserved in textual embeddings. The following section, therefore, reviews embedding inversion attacks and the threat models under which they have been studied.

2.6.1 Embedding Inversion Attack

To formalise the embedding inversion problem, we adopt the notation of Huang et al. [27]. Let $x \in V^n$ denote a text sequence over a vocabulary V, and let $\phi : V^n \to \mathbb{R}^d$ be a text embedding encoder. The encoder maps the input text to a fixed-length vector representation $z = \phi(x) \in \mathbb{R}^d$. An embedding inversion attack aims to recover the original input x, or an approximation of it, from the observed embedding z. This can be described as learning a function f that approximates the inverse of the embedding encoder:

$$f(\phi(x)) \approx \phi^{-1}(\phi(x)) = x. \tag{2.1}$$

In practice, directly learning or computing such an inverse is difficult. Modern embedding models, particularly those based on deep neural architectures, do not admit a simple analytic inverse. Prior work, therefore, often treats inversion as a search or optimisation problem rather than as exact functional inversion. Since embedding models are trained to preserve semantic similarity, a candidate text that is semantically close to the original input is expected to produce an embedding close to the target vector. Following Morris et al. [3], the inversion objective can therefore be formulated as finding a candidate text $\hat{x}$ whose embedding is maximally similar to the observed embedding z:

$$\hat{x} = \arg\max_{x} \cos\left(\phi(x), z\right). \tag{2.2}$$

This approach differs from directly approximating ϕ^{-1} in Equation 2.1. Rather than requiring an explicit inverse function, the attacker only needs a mechanism for evaluating how close candidate embeddings are to the target embedding. Consequently, embedding inversion can be approached as a black-box optimisation or generation problem. However, even this formulation commonly assumes that the adversary can obtain embeddings for candidate texts, for example, by querying the target encoder.

Huang et al. [27] further distinguish inversion attacks according to the granularity of the reconstructed information. In token-level inversion, the adversary aims to recover individual tokens or keywords from the target embedding. In sentence-level inversion, the adversary attempts to reconstruct the full input sequence. Sentence-level reconstruction generally constitutes a stronger privacy threat, since it may reveal not only isolated semantic attributes but also the structure, phrasing, and contextual content of the original text.

The feasibility of these attacks depends strongly on the information available to the adversary. Prior work, therefore, considers several threat models, often characterised by the adversary's access to the embedding model ϕ. In a *white-box* setting, the attacker has access to the internal model's architecture and parameters. In a *black-box* query setting, the attacker cannot inspect the model internals but can submit input texts to ϕ and observe the corresponding embeddings. A more restrictive setting arises when the attacker has no query access to the target model and may not even know which embedding model was used to produce the observed vectors.

These distinctions are important because exact reconstruction is generally infeasible for modern embedding models. As noted by Morris et al. [3], embeddings produced by complex architectures such as transformers or recurrent neural networks do not provide a straightforward inverse mapping. Moreover, downstream applications often use representations extracted from higher layers of a model rather than lower-level features [5]. Such high-level representations may preserve broad semantic information while discarding lexical, syntactic, or ordering details, thereby making exact reconstruction more difficult. At the same time, the presence of paired text examples and their embeddings can provide valuable supervision for learning an approximate inverse mapping.

In addition to model access, prior work also varies in the amount and type of auxiliary data available to the attacker. Some approaches assume access to a paired dataset $D_L = \{(x, \phi(x))\}$ to train or adapt an inversion model. Others consider more limited settings in which only embeddings are available, requiring the attacker to infer text without direct supervision from corresponding input–embedding pairs. The following text reviews representative approaches under these different assumptions. An overview of these approaches is shown in Table 2.1.

White-box

Song and Raghunathan [5] focus on recovering a set of words from embeddings, without considering word order. In their white-box setting, the adversary has full access to the embedding model, including its architecture and parameters. Given a target embedding $z^* = \phi(x^*)$, the goal is to recover an input sequence $\hat{x}$ whose embedding closely matches z^*. This can be formulated as an optimisation problem that minimises the distance between the embeddings of the reconstructed and original inputs. However, since text is inherently discrete, direct optimisation over sequences is intractable. To address this, the authors utilise a continuous relaxation, where discrete token selection is replaced by a probability distribution over the vocabulary at each position. This relaxation is useful because it transforms the search over discrete token sequences into a differentiable optimisation problem. As a result, the adversary can use gradient-based algorithms to update the relaxed input representation rather than relying on exhaustive or combinatorial search over possible word sequences.

In practice, directly inverting high-level embeddings produced by deep models is challenging because they are abstract and compressed. To overcome this, a two-stage approach is employed: the target embedding is first mapped to a "lower-level" representation using a learned function M, and the reconstruction is then performed in this "lower-level" space, which retains more information about the original input. After optimisation, the final discrete sequence is obtained by selecting the most probable tokens. This approach allows the adversary to effectively approximate the original text, often recovering semantically similar or partially accurate reconstructions.

Black-box with query access

In the black-box setting considered by Song and Raghunathan [5], the adversary only has query access to the embedding model ϕ and cannot compute gradients with respect to the model parameters. As a result, direct gradient-based inversion is not applicable. The attack, therefore, posed as a supervised learning problem, in which an inversion model υ is trained to recover information directly from embeddings. Given an embedding $\phi(x)$, the model predicts the set of words $W(x)$ contained in the original input, without considering word order.

To train υ, the adversary uses an auxiliary dataset D_{aux} and queries the embedding model to obtain pairs $(\phi(x), W(x))$. The inversion model is then trained to maximise the likelihood of the correct word set given the embedding. This problem can be approached as a multi-label classification, where each word in the vocabulary is independently predicted as present or absent, or as a multi-set prediction task, where words are generated sequentially con-

ditioned on previously predicted words. The latter approach captures word dependencies and has been shown to achieve better performance.

The authors of Vec2Text [3] consider a black-box setting, where the adversary can query the embedding model ϕ but does not have access to its parameters or gradients. The goal is to recover a text $\hat{x}$ whose embedding closely matches a target embedding $z = \phi(x^*)$, i.e., such that $\cos(\phi(\hat{x}), z)$ is maximised. While multiple texts may map to similar embeddings, the authors assume that such collisions are sufficiently rare in practice.

Since enumerating all possible text sequences is computationally infeasible, the problem is framed as learning a conditional distribution of text given embeddings. In particular, Vec2Text trains a model to generate text from embeddings and iteratively refines an initial hypothesis. Starting from an initial guess $x^{(0)}$, the model produces a sequence of hypotheses $x^{(0)}, x^{(1)}, \ldots$, where each subsequent hypothesis $x^{(t+1)}$ is a correction of the previous one and is expected to be closer to the target embedding in the vector space. This iterative refinement process allows the model to progressively improve the reconstruction by leveraging feedback from the embedding space.

Since most prior approaches have primarily been evaluated on English embeddings, Chen et al. [28] focused on identifying how the original language affects the recoverability of the original text. They considered the black-box-with-known-model scenario, where both the embeddings and API access were obtained, and utilised the Vec2Text [3] solution in a multilingual setting. Their aim was to invert sentence embeddings created by a multilingual model.

Black-box with input data

Huang et al. [27] aim to reconstruct text from embeddings, considering a scenario where a dataset D_L is leaked, containing both the original text and its embeddings from an unknown model. Their approach to reconstructing the unknown model is to train an encoder to map text from D_L to embeddings that best match the original ones. After training the encoder and minimising the loss between the original embeddings and the newly created embeddings to the extent that the two embeddings are in the same distribution, a decoder is trained to convert the embeddings back to the original text. This enables the adversary to reconstruct future leaked embeddings produced by the same source model.

Chen et al. [29] propose an alternative inversion approach that reduces the dependence on large amounts of paired data. Their method is based on a pre-trained encoder-decoder model, trained to map text to embeddings and reconstruct the original text from those embeddings.

To apply this model to a victim embedding space, the authors introduce an embedding alignment module that maps the victim embeddings to the representation space of the pre-trained model. This allows the pre-trained

decoder to be reused for reconstruction, even when the original embedding model differs from the one used during pre-training.

To learn the alignment, the adversary assumes access to a small number of paired samples $(x, \phi(x))$ from the victim model. Notably, the authors show that only a very limited number of such pairs is required: a single pair can already enable partial reconstruction, while around 30 pairs are sufficient to achieve meaningful recovery performance (e.g., ROUGE-L[6] above 20).

Overall, this approach differs from prior methods such as Vec2Text [3] by separating the reconstruction problem into alignment and decoding stages, thereby significantly reducing the required training data and improving reconstruction quality.

Taken together, these studies show that embeddings should not be treated as inherently private representations. Even when the original text is not directly available, embeddings may retain sufficient information to support various forms of inference or reconstruction. The feasibility and severity of such attacks depend on several factors, including the attacker's access to the embedding model, the availability of auxiliary text-embedding pairs, and the level of detail the attacker aims to recover. This motivates a closer examination of information leakage from textual embeddings in recommender-system settings, where item and user representations are often derived from semantically rich textual data.

2.6.2 Embedding Leaks in Recommenders

The literature has identified several vulnerabilities introduced by the use of language-model-based components in recommender systems. However, existing studies have often focused on manipulating recommendations or prompts, while paying less attention to privacy risks arising specifically from item embeddings produced by transformer-based text encoders. For example, Zhang et al. [31] build on the embedding inversion approach of Morris et al. [3] and show that, because LLM-based recommender systems rely heavily on item textual content, they can be vulnerable to subtle test-time modifications of item descriptions. Such modifications allow an attacker to increase the exposure of selected items while remaining difficult for platforms or users to detect.

Although several recent studies have investigated embedding inversion attacks, to the best of our knowledge, only limited prior work has examined embedding or representation inversion specifically in LLM-based recommender systems. Wang et al. [32] utilise the Morris et al. [3] approach and apply it specifically to LLM-based recommender systems. However, instead of inverting leaked embeddings, the authors experimented with inverting the output

[6]ROUGE-L measures the similarity between the ground-truth text x and the reconstructed text $\hat{x}$ by comparing their longest common subsequence, thereby capturing the degree of textual overlap between the two sequences [30].

logits and the attacker's ability to reconstruct the original prompt, including the items incorporated into it. Wang et al. [32] show that an attacker does not necessarily need access to prompt embeddings. If the attacker can obtain output logits and query the victim recommender system, they can issue a large number of queries using fabricated inputs from publicly available datasets. The resulting input-logit pairs are then used to train an inversion model that maps logits back to possible textual prompts. The differentiating factor of their work is that, unlike a typical representation, they do not use item embeddings; instead, they use the model's output logits. Nevertheless, the work is relevant because it shows that textual information used in LLM-powered recommender systems can be recovered from intermediate or output representations.

From Embedding Inversion to Item Identification

Recent work has questioned the assumption that embeddings produced by language-model-based text encoders are inherently privacy-preserving. In particular, *embedding inversion attacks* have shown that, under certain conditions, it is possible to reconstruct the original input text from its vector representation. These approaches aim to recover a text $\hat{x}$ such that its embedding $\phi(\hat{x})$ is close to a given target embedding z, thereby revealing sensitive information that was assumed to be hidden.

The relevance of embedding inversion attacks to this work lies in the shared threat model. Both lines of research assume a scenario in which an attacker gains access to embeddings, for example, through data sharing with a third-party recommender system provider. While such embeddings are often treated as anonymised or non-interpretable representations, prior work has shown that they may still encode substantial information about the original input.

However, the objective of this work differs from full inversion. Instead of directly reconstructing the original text, we investigate whether embeddings can be linked back to their corresponding items without knowing the exact textual features specific to recommender systems. Considering different information available to the attacker, we transform the task from inversion to embedding *matching* or *retrieval*, where the goal is to identify which item corresponds to a given embedding.

These two perspectives can be understood as complementary points on a spectrum of information leakage. Embedding inversion attacks represent a stronger form of leakage, as they attempt to recover the full textual content. In contrast, the approach considered in this work represents a more constrained but practically relevant scenario in which the attacker exploits the structure of the embedding space to perform linkage attacks.

Importantly, successful matching already implies a significant privacy risk. If an embedding can be reliably associated with a specific item, the attacker can indirectly recover the item's content, even without reconstructing the original

Table 2.1 Overview of Prior Embedding Attack Methods

Work	Attack Type	Threat Model	Data Assumptions
[5]	Attribute inference, membership inference, inversion (token-level)	White-box / Black-box (query)	Auxiliary dataset or full model access
[3]	Embedding inversion (sentence-level)	Black-box (query access)	Access to multilingual embeddings and query interface
[28]	Embedding inversion (multilingual)	Black-box	Access to multilingual embeddings and query interface
[27]	Embedding inversion (sentence-level)	Black-box	Paired dataset $D_L = \{(x, \phi(x))\}$
[29]	Embedding inversion (sentence-level)	Black-box (limited paired data)	Small number of paired samples
[32]	Representation inversion (logit-based)	Black-box (query access to recommender system)	Synthetic query-logit pairs

text. This is particularly relevant in recommender systems, where item catalogues are often publicly available or can be approximated. Therefore, demonstrating the feasibility of such matching attacks provides further evidence that embeddings should not be considered inherently secure representations.

In summary, embedding inversion attacks motivate the investigation conducted in this work by highlighting the potential for information leakage in transformer-based textual embeddings. Building on this insight, we focus on a practical attack scenario tailored to recommender systems, in which leaked embeddings are exploited to identify the corresponding items.

Chapter 3

Methodology

This chapter introduces the threat model for item identification from embeddings and outlines the proposed attack framework used to evaluate the described scenarios. The threat model defines the different levels of information available to an attacker, while the proposed attack framework describes how these assumptions are incorporated into a two-stage attack pipeline, consisting of embedding model identification followed by item identification.

3.1 Problem Definition

We consider a recommender system in which users interact with items, such as movies, books, or products. Let $\mathcal{U}$ denote the set of users and $\mathcal{I}$ the set of items. The system observes user-item interactions and aims to provide personalised recommendations based on this data. In modern recommender systems, additional semantic information about items is often incorporated to improve recommendation quality. Due to the recent success of large language models, this type of information is often derived from textual descriptions and encoded using sentence transformers or other embedding models. In the thesis, we denote ϕ as such an embedding model, which maps an (natural language) item description x_i to a dense vector representation $z_i = \phi(x_i) \in \mathbb{R}^d$.

In practical scenarios, a recommender system provider may wish to share interaction information and item representations with external parties, such as business collaborators or the research community. Typically, they do not provide explicit user or item identifiers; instead, they use anonymised IDs that do not directly identify users or items. Denote by $y_i \in Y$ the label of item i that is a unique piece of information that identifies an item, for instance, the title of a movie. Moreover, rather than releasing raw textual data or explicitly identifiable item information, the provider shares embeddings $z_i \in Z$, assuming that these representations are not human-readable and therefore preserve the privacy of the underlying data. In simpler terms, accessing z_i does not reveal

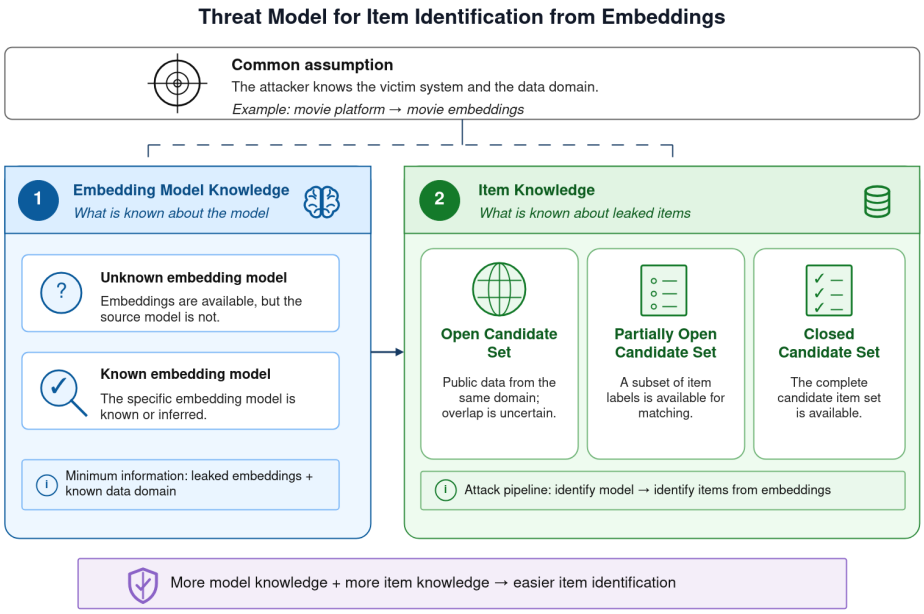

Figure 3.1 Threat Model for Item Identification from Embeddings

x_i (thus y_i), leaving the item labels unknown to the observer in the absence of an attack. Thus, this approach is often considered a form of privacy-preserving data sharing, enabling downstream tasks such as model training or analysis without exposing sensitive content.

In this work, we aim to systematically evaluate the robustness of embedding-based data sharing in recommender systems. Specifically, we propose a methodology to assess whether items can be identified from their embeddings under realistic attacker assumptions. By analysing different levels of attacker knowledge, we investigate the extent to which embeddings leak information and whether they can be considered a reliable mechanism for protecting item identity.

3.2 Threat Model for Item Identification from Embeddings

Attackers' ability to identify items from leaked embeddings depends strongly on the information available to them. We focus exclusively on black-box settings, in which the attacker has no access to model internals, such as gradients or training data, but may possess varying degrees of auxiliary information, including access to the model weights and architecture used to generate the victim's embeddings.

In all considered scenarios, we assume that the *attacker knows* the identity of the *victim system* and, consequently, the general *data domain*. For exam-

ple, if the victim is a movie streaming platform, the attacker can reasonably assume that the embeddings are generated from movie data. This assumption reflects realistic conditions in recommender systems datasets, where the application domain is often publicly known. The explored threat model is shown in Figure 3.1 and described in greater detail below.

(1) Identifying the Embedding Model. A key factor influencing the feasibility of downstream attacks is whether the attacker knows which embedding model was used to transform raw text x_i into representations z_i. We consider two attack scenarios, depending on whether the attacker knows the embedding model in advance:

- **Known embedding model.** The attacker has access to the embeddings z_i and knows in advance which LLM or sentence transformer (ϕ) was used to generate them.
- **Unknown embedding model.** The attacker has access to the embeddings z_i but *does not* know which LLM or sentence transformer (ϕ) was used to generate them. In this case, the attacker may first attempt to identify the embedding model before performing subsequent attacks.

Both scenarios are realistic: in some cases, the embedding model may be documented, standardised, or leaked, while in others it remains hidden but may still be inferred from the geometry and dimensionality of the released embeddings before performing item identification. Note, however, that the latter case is more challenging.

(2) Identifying Items from Embeddings. Given access to the leaked embeddings z_i, and potentially knowledge of the embedding model ϕ, the attacker's goal is to identify the underlying items. We always assume that the attacker never observes the labels y_i of item i in the victim set $\mathcal{I}$. However, the attacker may observe a candidate set $\bar{\mathcal{I}}$ such that $\mathcal{I} \cap \bar{\mathcal{I}} \neq \emptyset$, making the labels $\bar{y}_j$ available for all $j \in \bar{\mathcal{I}}$. Recall also that, for each victim item $i \in \mathcal{I}$, the attacker observes only its embedding $z_i = \phi(x_i)$, while the original item description x_i remains hidden. In contrast, for each candidate item $j \in \bar{\mathcal{I}}$, the attacker has access to a textual description $\bar{x}_j$, obtained, for example, by crawling or generation. The attacker's goal is to recover a mapping

$$\rho : \mathcal{I} \to \bar{\mathcal{I}},$$

such that $\rho(i) = j$ when the leaked embedding z_i corresponds to the candidate label $\bar{y}_j$.
Remark: By corresponding, we do not mean that $z_i = \phi(\bar{x}_j)$ exactly, but rather that both representations refer to the same underlying item ($y_i = \bar{y}_j$).

This work focuses on the following scenarios:

- **Closed candidate set.** The attacker has access to the complete candidate catalogue, and this catalogue coincides with the victim provider's item set, i.e., $\mathcal{I} = \bar{\mathcal{I}}$. However, the attacker does not know the labels y_i and the mapping ρ between the leaked embeddings and the corresponding item identities.
- **Partially open candidate set.** The attacker has access to a partial candidate catalogue that overlaps with the victim provider's item set, i.e., $\bar{\mathcal{I}} \cap \mathcal{I} = \bar{\mathcal{I}}$. In this case, the attacker observes labels $\{\bar{y}_j\}_{j \in \bar{\mathcal{I}}} \subset Y$ corresponding to a subset of items, but does not know the mapping ρ between the leaked embeddings and the corresponding item identities.
- **Open candidate set.** The attacker has access to an external candidate catalogue that strictly contains the victim provider's item set, i.e., $\mathcal{I} \subsetneq \bar{\mathcal{I}}$. In this case, the attacker does not know which items in $\bar{\mathcal{I}}$ belong to $\mathcal{I}$, nor the mapping ρ between leaked embeddings and item identities.

These scenarios are realistic because attackers may obtain item information through web crawling or public sources, ranging from complete coverage (closed set), to partial overlap (partially open set), to large external catalogues (such as IMDb for the movie domain) that contain many additional items beyond those of the victim (open set), while the correspondence to embeddings remains unknown. Moreover, they reflect varying levels of difficulty for the attacker. In particular, the availability of a candidate item set transforms the problem into a matching or retrieval task, while the absence of such information requires more general inference.

The combination of embedding model knowledge and item knowledge defines a spectrum of attack scenarios, ranging from highly constrained settings with minimal prior information to more informed settings where identification becomes significantly easier. This threat model provides a structured framework for evaluating the extent to which embeddings leak information and for understanding the practical risks associated with their use in recommender systems.

3.3 Proposed Attack Framework

Building on the taxonomy of item identifiability introduced in the previous section, we now formalise the attack pipeline. The proposed approach consists of two stages. First, the attacker identifies the embedding model used to generate the leaked representations. Second, given this model, the attacker attempts to recover the underlying items from the embeddings under different assumptions about the available auxiliary information.

3.3.1 Embedding Model Identification

We first consider the problem of identifying the embedding model used to generate the leaked victim embeddings. Let $\phi^\star$ denote the unknown embedding model used by the victim, and let

$$z_i = \phi^\star(x_i), \quad z_i \in \mathbb{R}^d,$$

be the observed embeddings corresponding to unknown input texts x_i drawn from a known domain.

We assume the attacker has access to the full set of victim embeddings $\{z_i\}$, but not to the original texts x_i nor the model $\phi^\star$. However, the dimensionality d of the embeddings is observable, which allows the attacker to restrict the set of candidate models[1] to

$$\Phi_d = \{\phi \mid \phi : \mathcal{X} \to \mathbb{R}^d\}.$$

where $\mathcal{X}$ denotes the space of textual item descriptions.

To identify $\phi^\star$, the attacker leverages auxiliary data from a publicly available dataset of the same domain. Let $\bar{\mathcal{X}} = \{\bar{x}_j\}$ denote a set of attacker-controlled texts sampled from the recommender domain. For each candidate model $\phi \in \Phi_d$, the attacker computes embeddings

$$\bar{z}_j = \phi(\bar{x}_j).$$

The key intuition is that embedding models induce distinct geometric structures in the vector space. If $\phi = \phi^\star$, then the distributions of $\bar{z}_j$ (derived from $\bar{\mathcal{I}}$) and z_i (derived from $\mathcal{I}$) should be aligned, whereas embeddings generated by different models will exhibit structural discrepancies.

To formalise this, we define a reconstruction-based criterion. For each candidate model $\phi \in \Phi_d$, an autoencoder f_ϕ is trained on the observed attacker embeddings $\bar{z}_j$. The trained model is then used to reconstruct the victim embeddings z_i, and the reconstruction error is measured as

$$\mathcal{L}(\phi) = \frac{1}{|Z|} \sum_{z_i \in Z} \|z_i - f_\phi(z_i)\|_2^2,$$

where Z denotes the set of victim embeddings.

▶ Hypothesis 3.3.1. The reconstruction loss $\mathcal{L}(\phi)$ is minimised when $\phi = \phi^\star$.

Under this hypothesis, the attacker can estimate the victim's embedding model as

$$\hat{\phi} = \arg\min_{\phi \in \Phi_d} \mathcal{L}(\phi).$$

Note that we proposed a realistic attack scenario: even without access to the original data or the model internals, the attacker can exploit domain

[1] i.e., models that produce embeddings of the same dimensionality

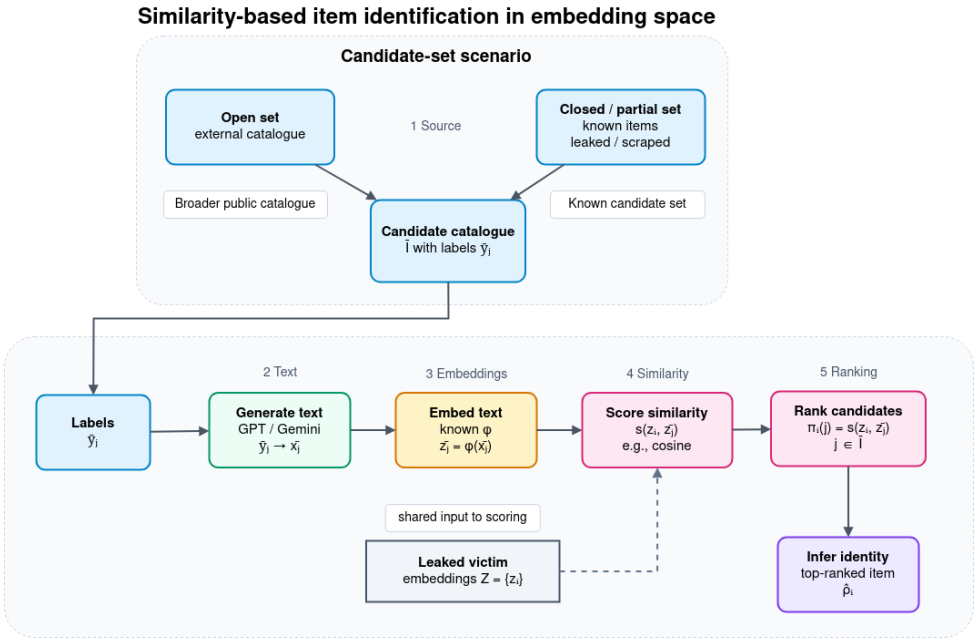

Figure 3.2 Similarity-based item identification in embedding space

knowledge and publicly available data to infer the embedding model. Consequently, the assumption that the embedding model is unknown should not be considered a strong defence, as it can be relaxed through a preliminary identification step.

3.3.2 Item Identification

Given the embedding model ϕ identified in the previous step, the attacker can construct embeddings comparable to those of candidate items. This enables formulating the item identification task as a similarity-based matching problem in the embedding space. Depending on the information available to the attacker, we distinguish between the following two scenarios as shown in Figure 3.2.

Closed candidate set / Partially open candidate set We first consider the scenario in which the attacker aims to match leaked victim embeddings Z to a set of known candidate items $\bar{\mathcal{I}}$ (with $\bar{\mathcal{I}} \cap \mathcal{I} = \bar{\mathcal{I}}$). Observe that these known items can result not only from information leakage from the recommender system provider but also from online extraction from the victim's website.

Given the known labels $\bar{y}_j$ corresponding to items in $\bar{\mathcal{I}}$, we assume the attacker (1) generates textual descriptions $\bar{x}_j$ using a generative model, such as

GPT or Gemini. After that, they (2) use the embedding model ϕ to transform these generated descriptions into embeddings,

$$\bar{z}_j = \phi(\bar{x}_j).$$

The attacker then (3) defines a similarity function $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, such as cosine similarity. Finally, for each victim embedding z_i, the attacker (4) ranks all candidate items $j \in \bar{\mathcal{I}}$ according to their similarity scores:

$$\pi_i(j) = s\big(z_i, \bar{z}_j\big), \qquad j \in \bar{\mathcal{I}}.$$

This induces a ranking $\hat{\rho}_i$ over the candidate catalogue $\bar{\mathcal{I}}$, where candidates with higher similarity scores are considered more likely to correspond to the victim item i. The resulting rankings are then used to infer the correspondence between victim embeddings and item identities.

Open Candidate Set. We now consider a more challenging scenario in which the attacker does not have access to the exact set of items used to generate the victim embeddings. However, we assume that the attacker has access to a (possibly publicly available) dataset containing all items in the recommender system, i.e., $\mathcal{I} \subseteq \bar{\mathcal{I}}$.

In this setting, the method begins with the attacker (1) collecting the external candidate catalogue $\bar{\mathcal{I}}$ and its labels $\{\bar{y}_j\}_{j \in \bar{\mathcal{I}}}$ from an available dataset. Unlike the closed-set case, this catalogue is not assumed to have been leaked by the victim provider, but rather obtained from public or auxiliary sources. The remaining steps follow the same procedure: (2) generate textual descriptions $\bar{x}_j$ from labels $\bar{y}_j$ using a generative model; (3) use the embedding model ϕ to transform these generated descriptions into embeddings $\bar{z}_j = \phi(\bar{x}_j)$; (4) define a similarity function s; and (5) rank all candidate items $j \in \bar{\mathcal{I}}$ according to their similarity scores with each victim embedding.

Chapter 4

Experiments

This chapter discusses experiments which were conducted for each of the selected approaches from Section 3.3. In addition, this chapter describes selected datasets, embedding models, and other implementation details specific to these experiments, along with the final results of each approach.

4.1 Selected Models

Textual inputs were converted into dense vector representations using pre-trained sentence transformer models from Hugging Face. Sentence transformers are appropriate for this purpose because they are based on the transformer architecture, which underpins modern large language models. Transformer-based models encode text by applying self-attention to input tokens, producing contextual hidden representations. Sentence transformer models adapt this mechanism for sentence or document-level representation learning by pooling token-level representations and training or fine-tuning the resulting embeddings so that semantically similar texts are mapped to nearby points in the vector space. Consequently, they provide fixed-size embeddings for semantic comparison, retrieval, clustering, and downstream attack evaluation.

In this work, sentence transformer models serve as a practical approximation of embedding generation in modern language-based systems. Although sentence transformer models and large language models may differ in scale, training objectives, and deployment settings, both rely on transformer-based text representations. Therefore, evaluating attacks on sentence transformer embeddings provides insight into the risks associated with storing, indexing, or exposing semantic embeddings more generally. In addition, sentence transformers are also commonly used in academic and real-world scenarios to generate embeddings for recommender systems [33, 34, 35].

To evaluate attacks across different model families and embedding dimensions, six pre-trained sentence transformer models were selected. The selection

includes three models with 384-dimensional embeddings and three models with 768-dimensional embeddings. This design enables assessing whether attack performance is affected by differences in embedding size, model architecture, and training objective, rather than being tied to a single representation space.

The selected sentence transformer models [36] are shown in Table 4.1.

384-dimensional embeddings	768-dimensional embeddings
`all-MiniLM-L6-v2`	`all-distilroberta-v1`
`paraphrase-MiniLM-L3-v2`	`multi-qa-mpnet-base-dot-v1`
`paraphrase-multilingual-MiniLM-L12-v2`	`paraphrase-albert-small-v2`

Table 4.1 Selected sentence transformer models grouped by embedding dimensionality.

The selected models cover several common sentence transformer families. The `all-*` models are general-purpose embedding models trained on large collections of sentence pairs. They are intended to perform well across a broad range of tasks, including semantic textual similarity.

The `paraphrase-*` models are optimised for capturing semantic equivalence between texts and are commonly used for semantic similarity tasks. Their inclusion allows the evaluation to consider models whose training objectives are explicitly aligned with identifying semantically similar textual inputs.

The `multi-qa-mpnet-base-dot-v1` model represents a semantic search-oriented model family. It is trained on large-scale question-answer data and is designed to match queries with relevant passages. Including this model broadens the evaluation beyond general-purpose and paraphrase-oriented embeddings by incorporating a model optimised for retrieval-style tasks.

Overall, the selected sentence transformer models differ in dimensionality, architecture, computational cost, and training objective. This diversity enables a more comprehensive analysis of how different embedding spaces affect item identification and embedding model identification.

For synthetic description generation, the `gpt-oss-20b` model [37, 38] was used. This model is an open-weight large language model released by OpenAI under the Apache 2.0 license. It contains approximately 21 billion parameters and is based on a Mixture-of-Experts architecture, in which only a subset of parameters is active during each forward pass. This design reduces computational requirements compared with dense models of similar total parameter count, enabling more efficient deployment. In this work, the model is used solely to generate auxiliary item descriptions from titles, thereby constructing attacker-side textual inputs for the embedding model identification setup [37].

4.2 Datasets

This section presents the datasets used to evaluate the proposed approaches of embedding inversion attacks in recommender systems. To study this threat,

the selected datasets cover two domains, specifically movies and books. The primary datasets (MovieLens and Goodreads) are used to generate and evaluate embeddings under realistic recommendation scenarios, while additional datasets (TMDB and Amazon Reviews) serve as external or adversarial sources that simulate the knowledge available to an attacker. This combination enables a comprehensive analysis of how embedding representations encode information.

MovieLens 32M The MovieLens 32M dataset [39] is a large-scale benchmark dataset for recommender systems provided by the GroupLens Research Lab at the University of Minnesota. It contains 32,000,204 user ratings and 2,000,072 tag applications across 87,585 movies, generated by 200,948 users between 1995 and 2023 [40]. Ratings are provided on a 5-star scale with half-star increments, and each user has rated at least 20 movies [40].

The dataset captures both explicit feedback (ratings) and implicit semantic information (user-generated tags), making it suitable for evaluating collaborative filtering and hybrid recommender systems. MovieLens datasets are widely used as standard benchmarks in recommender systems research [41].

Extended MovieLens 32M The extended MovieLens 32M dataset [42] augments the original MovieLens [39] data with additional textual and semantic metadata, such as movie descriptions or semantic identifiers. While the original dataset primarily focuses on user-item interactions, the extended version enables integrating content-based features into recommendation models.

Such extensions are commonly used in recommender systems research and enable a more complex integration of implicit semantic information from item-based features.

Goodreads Dataset The Goodreads Book Graph dataset [43, 44] is a large-scale dataset from 2017 for book recommendation and analysis. It consists of three main components: (1) book metadata, (2) user-book interactions, and (3) user-generated reviews [45].

The dataset contains over 2.3 million books, 876,145 users, and approximately 229 million user-book interactions [45]. Additionally, it includes millions of textual reviews, enabling NLP tasks.

Files such as `goodreads_interactions.csv` store interaction data, while `goodreads_books.jsonl` provides detailed metadata for each book. Although much more information can be extracted from the Goodreads dataset, in this thesis, we focused only on the information available in these two files.

TMDB Movies Dataset The TMDB [46] dataset is a large collection of movie metadata derived from The Movie Database (TMDb), a community-driven platform for movie information. The dataset contains information about

over one million movies, including attributes such as titles, genres, release dates, and ratings [47].

A key characteristic of this dataset is that it is updated regularly (reported daily). We accessed this dataset on 20 February 2026.

Unlike MovieLens, which focuses on user interactions, TMDB primarily provides rich item metadata, making it suitable for adversarial scenarios that require external knowledge about items.

Amazon Reviews 2023 Dataset The Amazon Reviews 2023 dataset [48], provided by McAuley Lab, is a large-scale e-commerce dataset containing product metadata, user reviews, and ratings across multiple domains, including books [49].

These datasets are widely used as benchmarks in recommender systems research because they combine structured interaction data with unstructured textual information. The 2023 version extends previous releases by providing richer metadata and multimodal features, making it suitable for modern approaches that combine textual and structured item information [49].

4.2.1 Dataset Construction and Preprocessing

The experimental setup supports two related evaluation tasks: item identification and embedding model identification. Both tasks are based on the assumption that an adversary observes embeddings generated from unknown textual inputs and attempts to infer information about the embedded items or the embedding model used to produce them. The datasets are therefore constructed to provide victim-side and attacker-side textual corpora under controlled assumptions.

For the victim-side data, the MovieLens and Goodreads datasets are used to represent the movie and book domains, respectively. Depending on the experiment, item text is represented using one of several textual input formats:

- **`title: description`**: the concatenation of an item title and its original description.
- **`title`**: the item title alone.
- **`description`**: the original item description alone.
- **`generated description`**: a synthetic summary generated from the item title using the `gpt-oss-20b` model.

Goodreads already provides rich textual descriptions for book items. Since the original MovieLens dataset lacks sufficiently detailed descriptions, an extended MovieLens version enriched with textual descriptions from Hugging Face [42] is used. This enriched metadata is merged with the original MovieLens interaction data in order to preserve item-level interaction information.

The construction of the attacker-side data differs between the two evaluation tasks. For item identification, the attacker-side and victim-side corpora must contain mutual items, since the goal is to identify which known item corresponds to a given victim embedding. Therefore, MovieLens and Goodreads are used on both sides of the item identification setup. This allows the adversary to construct embeddings for candidate items and compare them against the victim's embeddings, assuming the attacker has access to some item metadata for the same item universe.

For embedding model identification, mutual items between the victim and attacker corpora are *not required*. On the contrary, the goal is to learn model-specific properties of embedding spaces rather than explore common items. In this setting, external domain datasets are used to train the autoencoders used for embedding model identification, creating a realistic scenario in which an attacker lacks access to the exact victim dataset and instead uses an external dataset. Specifically, TMDB is used as the auxiliary movie-domain dataset, and Amazon Reviews as the auxiliary book-domain dataset. These datasets serve as external attacker-side knowledge sources from which domain-relevant textual inputs can be generated without assuming access to the victim's exact item set.

For the TMDB and Amazon Reviews datasets, only item titles are used as the initial source of information. Since the attacker is not assumed to know the victim's original descriptions, synthetic summaries are generated for each selected title using the `gpt-oss-20b` model. As the attacker lacks information about which features were embedded by the victim, only the resulting `"descriptions"` form an auxiliary corpus for training the autoencoders used in embedding model identification. This procedure simulates a scenario in which the attacker has general domain knowledge but lacks the original textual inputs embedded by the victim.

During preprocessing, entries with missing, empty, or non-informative descriptions are removed. This includes placeholder descriptions indicating that no summary is available. Although not all entries are manually inspected, this filtering step reduces the presence of low-information texts and ensures that the resulting embeddings contain meaningful semantic content. This is important for both evaluation tasks, since item identification depends on item-specific textual information, while embedding model identification depends on consistent semantic representations across models.

All datasets are restricted to English-language items. Language filtering is performed using the `lid.176.bin` model [50, 51] from fastText [52], together with verification that the text contains Latin characters. For the TMDB dataset, the `original_language` field is also used, and only entries labelled as `en` are retained. The language identification model's confidence scores are not used as an additional filtering criterion; texts are retained if they are classified as English.

For the Extended MovieLens and Goodreads datasets, both titles and descriptions are checked for English-language consistency. For the TMDB and Amazon Reviews auxiliary datasets, only titles are checked before summary generation, since the original descriptions are not used. During summary generation, the `gpt-oss-20b` model is explicitly instructed to produce summaries in English.

Additional preprocessing is applied to ensure the validity of the generated adversarial corpora. For example, movie titles associated with adult content are removed to satisfy the language model's generation constraints. Since the model may lack sufficient knowledge of all items, only titles for which a valid description is successfully generated are retained. For both the movie and book domains, 10,000 valid English-language `"descriptions"` are selected from the successfully processed adversarial items.

The resulting victim-side and attacker-side textual inputs are encoded using the selected sentence transformer models. For item identification, these embeddings are used to compare victim embeddings against attacker-generated candidate embeddings for the same item domain. For embedding model identification, embeddings derived from the TMDB and Amazon auxiliary corpora are used to train autoencoders that capture model-specific properties of the embedding spaces.

Due to the size of the datasets, the experiments are restricted to the 10,000 most frequently interacted items in each victim dataset. This selection reflects a realistic recommendation setting, where popular items dominate user interactions and are therefore more likely to be embedded, indexed, and exposed in practical systems.

After preprocessing the datasets, 10,000 item titles from TMDB and Amazon Books were randomly selected to simulate an attacker's dataset. The titles of the attacker's and victim's datasets were compared to determine the extent to which the number of common items affects the results.

The attacker and the victim shared 332 movie titles and 41 book titles. This result is due to the random selection of the attacker's dataset, whereas the victim's dataset was selected only for the top 10,000 most interacted items.

4.3 Evaluation Metrics

The task is formulated as an embedding matching problem. For each victim embedding x_j, its similarity to all attacker-generated embeddings $\bar{x}_j$ is computed, producing a ranking over candidate items. The objective is to correctly identify which item corresponds to each victim embedding.

Hit Rate (HR@1). Hit Rate at 1 measures the proportion of cases in which the correct item is ranked as the most similar candidate:

$$\text{HR@1} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\text{rank}(i) = 1)$$

where $\text{rank}(i)$ denotes the position of the correct item in the ranked list for x_i, and $\mathbf{1}(\cdot)$ is the indicator function. This metric corresponds to exact identification accuracy. Hit@1 is the primary metric because it captures the exact item identification. In this setting, an attack is most successful when the correct candidate item is ranked first for a given victim embedding. A high Hit@1 therefore indicates that the embedding contains enough information to directly reveal the item identity.

Mean Reciprocal Rank (MRR). Mean Reciprocal Rank evaluates how highly the correct item is ranked:

$$\text{MRR} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{\text{rank}(i)}$$

Higher values indicate that correct matches tend to appear near the top of the ranking. MRR provides additional information about whether the correct item appears near the top of the ranking.

Mean Squared Error (MSE) For training the dense autoencoder, the Mean Squared Error loss was used. In PyTorch, this is implemented as `torch.nn.MSELoss`, which measures the average squared difference between the predicted output and the target values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$$

where x_i represents the reconstructed output and y_i the original input. The loss can also be interpreted as the squared L2 norm between corresponding elements of the input and target tensors [53]. By minimising this loss, an autoencoder is encouraged to reconstruct inputs as accurately as possible, making it a suitable objective for dense autoencoder training.

Root Mean Squared Error (RMSE) For the recommendation experiment, performance was evaluated using Root Mean Squared Error. RMSE is the square root of MSE and measures the average prediction error in the same scale as the original ratings:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{r}_i - r_i)^2}$$

where $\hat{r}_i$ is the predicted rating and r_i is the true rating. Lower RMSE values indicate more accurate rating predictions.

4.4 Implementation

Python. Python is a high-level, interpreted programming language with dynamic typing and an emphasis on code readability. It provides built-in data structures and supports multiple programming paradigms, including object-oriented programming [54]. Its simple and expressive syntax, together with an extensive standard library, makes it well-suited for rapid application development and scripting [54].

In addition, Python is widely used in machine learning and data processing workflows, largely due to its extensive ecosystem of libraries such as PyTorch and Transformers. This makes it well-suited for tasks involving embedding models, data preprocessing, and evaluation pipelines. And thus, it was chosen as the primary programming language of this work.

Used Libraries. The implementation relies on several Python libraries for data processing, modelling, and text handling.

PyTorch. The PyTorch library is used to build and train neural networks. Its `torch.utils.data.DataLoader` provides an iterable over datasets with support for batching and shuffling, while `TensorDataset` enables combining tensors into a dataset. The `torch.nn` module provides components for constructing neural network architectures [55].

Pandas. Pandas is a data analysis library providing the `DataFrame` structure, a two-dimensional, labelled tabular data structure widely used for data manipulation and preprocessing [56].

Random. The `random` module in Python's standard library is used to generate pseudo-random numbers and perform random sampling, which is commonly used for data splitting and shuffling.

fastText. fastText is an open-source library for efficient learning of word representations and text classification, capable of generating word vectors and performing language identification tasks [57].

unicodedata. The `unicodedata` module provides access to the Unicode Character Database and allows inspection and normalisation of character properties, which is useful for filtering text based on character sets (e.g., Latin script).

Transformers. The Hugging Face `Transformers` library provides a unified interface for state-of-the-art transformer-based models. Classes such as `AutoTokenizer` and `AutoModel` automatically load the appropriate tokeniser and model for a given pretrained architecture, while `AutoModelForCausalLM` is specialised for language modelling tasks [14].

re. The `re` module implements regular expressions, enabling pattern-based search, matching, and manipulation of text data, which is essential for text preprocessing and cleaning [58].

NumPy. NumPy provides efficient multidimensional array structures and numerical operations, forming the foundation for numerical computation in Python and enabling efficient data manipulation and indexing [59].

scikit-learn (sklearn). The `scikit-learn` library provides a wide range of tools for machine learning and data preprocessing. `train_test_split` function from `sklearn.model_selection` is used to divide datasets into training and validation subsets. This function splits input data into random train and validation portions while preserving the structure and type of the input arrays [60].

Splitting data into separate subsets is essential for unbiased model evaluation, as it ensures that performance is measured on data not seen during training. The `train_test_split` function also supports configurable parameters such as `test_size` for controlling the proportion of the dataset used for testing and `random_state` for reproducibility [60]. This makes it a suitable and efficient tool for preparing data in machine learning workflows.

4.5 Results

In this section, we will discuss the results of the approaches explored in Section 3.3, presenting experiments on identifying which embedding model was used and on handling both known and unknown items.

4.5.1 Embedding Model Identification

We first evaluate the embedding model identification stage introduced in Section 3.3.1. As mentioned in the Hypothesis 3.3.1, the correct embedding model should minimise the used recovery loss:

$$\hat{\phi} = \arg\min_{\phi \in \Phi_d} \mathcal{L}(\phi).$$

The purpose of the following experiments is therefore to test whether the reconstruction loss $\mathcal{L}(\phi)$ reliably identifies the true embedding model $\phi^\star$.

Experimental setup. To evaluate this hypothesis, we performed a grid search over autoencoder hyperparameters for each candidate sentence transformer. The searched parameters, shown in Table 4.2, are within the ranges for the latent dimensionality, learning rate, weight decay, and batch size. All autoencoders were trained using the Adam optimiser and MSE loss, with a maximum of 500 epochs and early stopping with a patience of 10 epochs. A fixed 80/20 train–validation split was used during hyperparameter selection.

Hyperparameter	Values
Latent dimensionality	$\{16, 32, 64, 128, 256\}$
Learning rate	$\{0.005, 0.01, 0.05, 0.1, 0.5\}$
Weight decay	$\{0, 10^{-6}, 10^{-5}, 10^{-4}\}$
Batch size	$\{256, 512, 1024, 2048\}$
Optimiser	Adam
Loss	Mean Squared Error
Maximum epochs	500
Early stopping patience	10 epochs
Train–validation split	80/20
Random seeds	100

Table 4.2 Autoencoder hyperparameter search space used for embedding model identification.

After selecting the best configuration for each candidate model, the corresponding autoencoder was trained across 100 random seeds in order to assess the stability of the reconstruction-based identification procedure.

Identification accuracy. Across all evaluated settings, the same pattern is observed: the autoencoder trained on embeddings produced by the same model as the victim embeddings obtains the lowest reconstruction loss. In other words, when the victim embeddings are generated by $\phi^\star$, the minimum of $\mathcal{L}(\phi)$ is consistently obtained for $\phi = \phi^\star$ as demonstrated in Table 4.3. This holds across both embedding dimensionalities, across the movies and books datasets, and for autoencoders trained with and without activation functions. These results support Hypothesis 3.3.1: the reconstruction loss provides a reliable signal for identifying the embedding model used to generate the leaked embeddings.

		Trained autoencoders		
		A	B	C
Test data	A	3.70 ± 0.01	34.40 ± 0.18	34.46 ± 1.71
	B	22.15 ± 0.09	0.86 ± 0.01	23.38 ± 1.25
	C	87.25 ± 0.34	89.92 ± 0.38	7.22 ± 2.73

Table 4.3 Cross-model recovery loss for MovieLens test embeddings reconstructed by autoencoders trained on TMDB movies. A: all-distilroberta-v1; B: multi-qa-mpnet-base-dot-v1; C: paraphrase-albert-small-v2

Effect of activation functions The clearest separation is obtained when autoencoders are trained without activation functions. In the 768-dimensional

book setting, the diagonal entries, corresponding to cases where the training and victim embeddings are generated by the same embedding model, remain consistently low. For example, the reconstruction losses are 3.70 ± 0.01 for `all-distilroberta-v1`, 0.86 ± 0.01 for `multi-qa-mpnet-base-dot-v1`, and 7.22 ± 2.73 for `paraphrase-albert-small-v2`. In contrast, off-diagonal evaluations, where the autoencoder is trained on one embedding space but tested on embeddings from another model, produce substantially larger losses, often above 20 and reaching up to 89.92. A similar pattern is observed in the book dataset: correctly matched models again produce low reconstruction losses, while mismatched embeddings produce substantially larger errors, frequently above 20.

The activation-based autoencoders preserve the same qualitative ordering but yield higher overall reconstruction losses. With activation functions, even correctly matched embeddings are reconstructed less accurately. In contrast to Table 4.3, losses of 20.18 for `all-distilroberta-v1` and 7.64 for `multi-qa-mpnet-base-dot-v1` are now observed in the movie setting. Mismatched embeddings remain substantially higher, exceeding 30 in several cases and reaching values above 115. Within these experiments, activation functions do *not* prevent model identification, since the correct model still obtains the lowest reconstruction loss, but they reduce reconstruction fidelity and make the separation between matched and mismatched cases less pronounced. For this reason, the non-activation setting provides clearer evidence for the proposed identification criterion.

Effect of domain shift. The results also suggest that the reconstruction loss is driven more by the geometry of the embedding space than by the semantic domain of the auxiliary data. When the autoencoder is trained and evaluated across different domains, for example, training on movie embeddings and evaluating on book embeddings, the loss for correctly matched embedding models increases only moderately. For instance, `multi-qa-mpnet-base-dot-v1` yields losses of 0.86 in the movies-on-movies setting, 1.30 in the movies-on-books setting, and 7.82 in the books-on-movies setting. Although these values show that domain shift can affect reconstruction quality, the effect is small compared with the increase caused by embedding-model mismatch. When training and evaluation use different embedding models, reconstruction losses remain much larger, regardless of whether the evaluation is within or across domains.

This distinction is important for the threat model. The attacker is not assumed to know the original victim texts x_i, but only to have access to auxiliary texts $\bar{\mathcal{X}}$ from a similar domain. The cross-domain results indicate that exact domain matching is not necessary for the reconstruction-based signal to remain useful. While using auxiliary data from the same domain can improve reconstruction quality, the dominant factor is whether the autoencoder f_ϕ was trained on embeddings produced by the same model as the victim embeddings.

This supports the underlying assumption of the proposed approach: different embedding models induce sufficiently distinct geometric structures that can be detected through reconstruction loss.

Stability across random seeds. The low variance across 100 random seeds further indicates that the observed behaviour is stable and not a consequence of a favourable initialisation. For correctly matched models, the reported standard deviations are very small, as shown in Table 4.3. The separation between matched and mismatched embeddings is also much larger than the variation across random seeds. Therefore, the identification result is not sensitive to a particular autoencoder initialisation or training run.

Key Takeaways

Overall, the experiments provide strong empirical support for the model identification stage of the proposed attack framework. For each candidate model $\phi \in \Phi_d$, the reconstruction loss $\mathcal{L}(\phi)$ captures how compatible the victim embeddings $\mathcal{Z}$ are with the embedding space learned from attacker-controlled auxiliary data $\bar{\mathcal{Z}}$. The correct model $\phi^\star$ consistently produces the lowest loss, allowing the attacker to estimate

$$\hat{\phi} = \arg\min_{\phi \in \Phi_d} \mathcal{L}(\phi)$$

with high reliability in the evaluated settings. These findings suggest that treating the embedding model as unknown does not by itself provide a strong defence against the subsequent item identification attack, since the model can be inferred from the leaked embeddings using only their dimensionality and auxiliary data.

4.5.2 Item Identification Results with Closed and Partially Open Candidate Sets

In this section, we study item identification introduced in Section 3.3 under both closed and partially open candidate-set scenarios. To evaluate how the choice of textual input affects identification performance, we vary both the victim-side text x_j and the attacker-side text $\bar{x}_j$. For the victim-side text, we consider two formats: (1) the concatenation of title and description, and (2) the description alone. For the attacker-side text, we consider: (1) the item title, and (2) a generated description. Each explored combination is repeated across 10 random seeds for various sizes of $|\mathcal{I}|$ and $|\bar{\mathcal{I}}|$ to assess the stability of this approach.

The results of varying victim and attacker combinations can be seen in Table 4.4, which summarises the main results at $|\bar{\mathcal{I}}| = 1{,}000$ and $|\mathcal{I}| = 10{,}000$. The full model-wise curves for the best embedding combinations are shown in

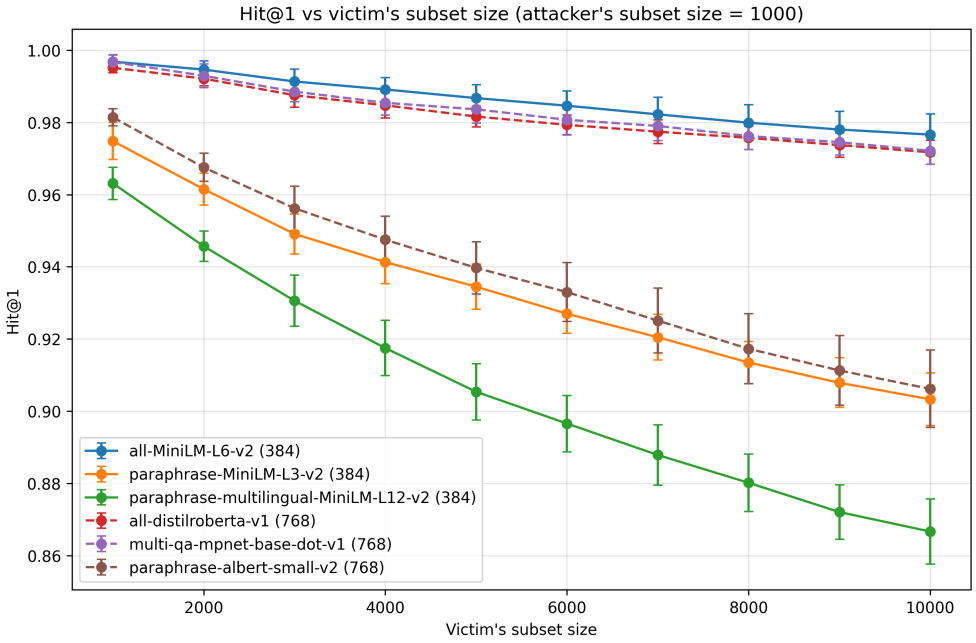

Figure 4.1 Hit@1 for MovieLens item identification with Victim: Title + Description / Attacker: Title, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

Figures 4.1 and 4.2 for the MovieLens dataset and Figures 4.3 and 4.4 for the Goodreads Books dataset. Results of the worst performing scenario are shown in Figures 4.5 and 4.6 for the MovieLens dataset and Figures 4.7 and 4.8 for the Goodreads dataset.

Figure 4.1 reports the Hit@1 performance of different sentence transformer models as the victim candidate-set size $|\mathcal{I}|$ increases, while the attacker subset size $|\bar{\mathcal{I}}|$ is fixed at 1,000 items. The first point of each sentence transformer corresponds to the closed-set setting, where $\mathcal{I} = \bar{\mathcal{I}}$. The remaining points correspond to the partially open setting described in Section 3.3, where the victim candidate set grows while the attacker set remains fixed.

Overall, Hit@1 decreases as $|\mathcal{I}|$ increases, showing that item identification becomes more difficult as the search space expands. However, this degradation is model-dependent. The models `all-MiniLM-L6-v2`, `all-distilroberta-v1`, and `multi-qa-mpnet-base-dot-v1` achieve the highest and most stable Hit@1 scores, with only a gradual decline as the victim subset size increases. In contrast, `paraphrase-MiniLM-L3-v2`, `paraphrase-albert-small-v2`, and especially `paraphrase-multilingual-MiniLM-L12-v2` exhibit a stronger performance drop, indicating lower robustness in larger candidate sets.

These results suggest that item-level distinguishability depends strongly on the embedding model. In particular, performance is not solely explained

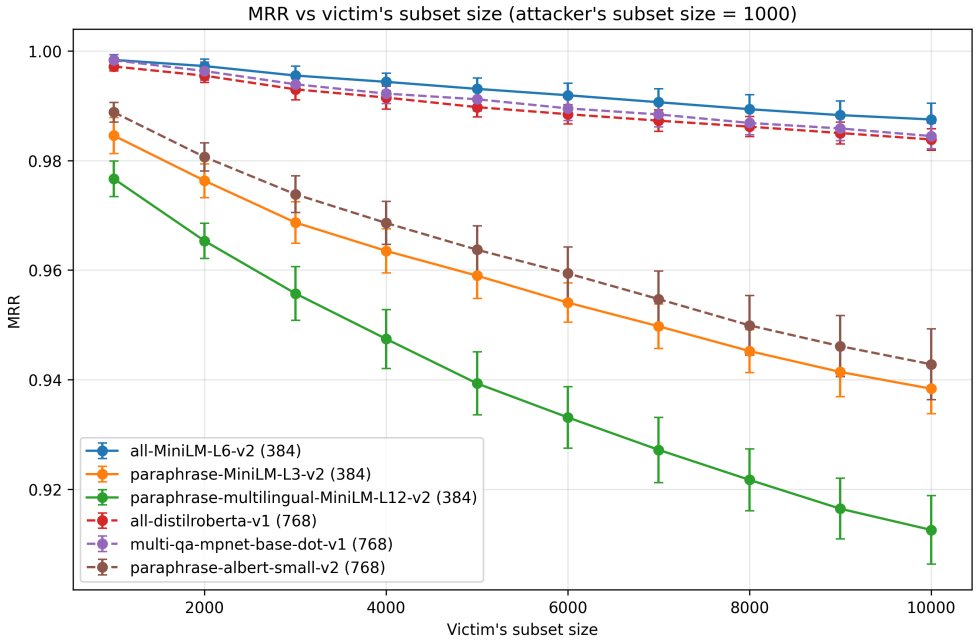

Figure 4.2 MRR for MovieLens item identification with Victim: Title + Description / Attacker: Title, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

by embedding dimensionality, since models with the same dimensionality can exhibit substantially different trends.

General Trends.

Across both datasets, increasing the victim subset size $|\mathcal{I}|$ consistently reduces Hit@1 and MRR. This is expected because a larger victim set increases the number of embeddings that must be matched, thereby increasing the likelihood of confusing semantically similar items. The decrease is gradual in both closed and partially open scenarios, indicating that the embedding space still preserves useful item-level structure even as the matching problem grows.

The results also show that MRR is consistently higher than Hit@1. Thus, even when the correct item is not retrieved as the nearest neighbour, it is often still ranked relatively high among the candidates. From a privacy perspective, this means that the attack may still substantially narrow the candidate set even when exact top-1 recovery fails.

In contrast, varying the attacker candidate subset size $|\bar{\mathcal{I}}|$ has a smaller effect than varying $|\mathcal{I}|$. For example, in the movie setting with x_j containing `title: description` and $\bar{x}_j$ containing `generated description`, increasing $|\bar{\mathcal{I}}|$ from 1,000 to nearly the full victim set $\mathcal{I}$ fixed at 10,000 produces only small changes in Hit@1 and MRR. This suggests that, in the evaluated setup,

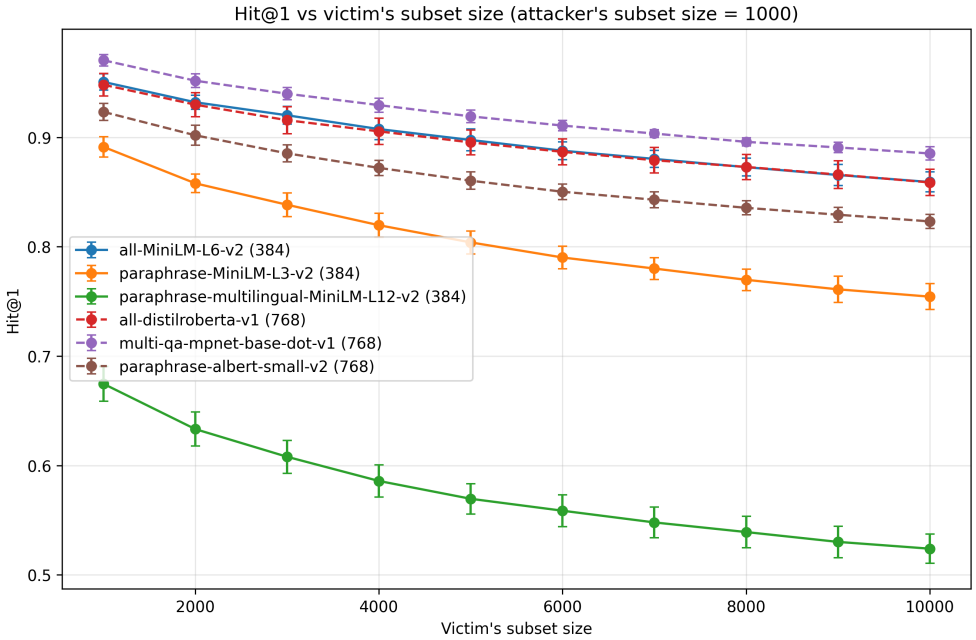

Figure 4.3 Hit@1 for Goodreads Books item identification with Victim: Title + Description / Attacker: Title, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

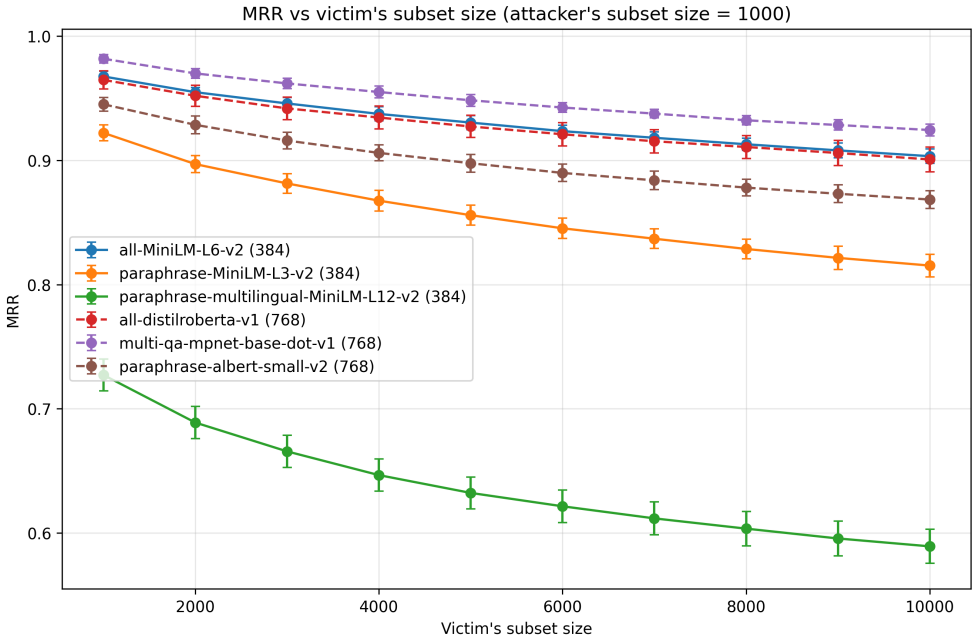

Figure 4.4 MRR for Goodreads Books item identification with Victim: Title + Description / Attacker: Title, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

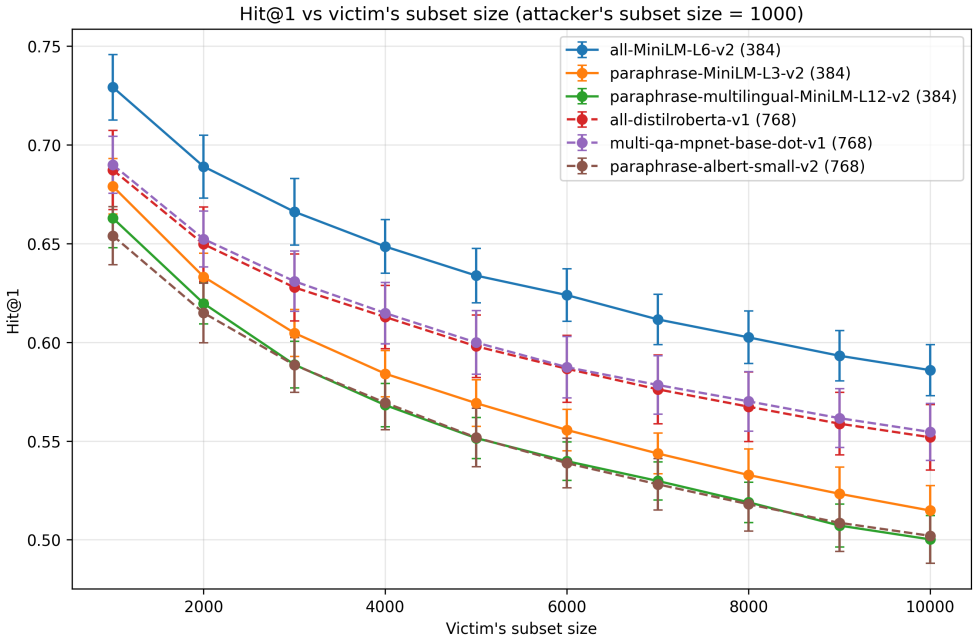

Figure 4.5 Hit@1 for MovieLens item identification with Victim: Description / Attacker: Generated Description, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

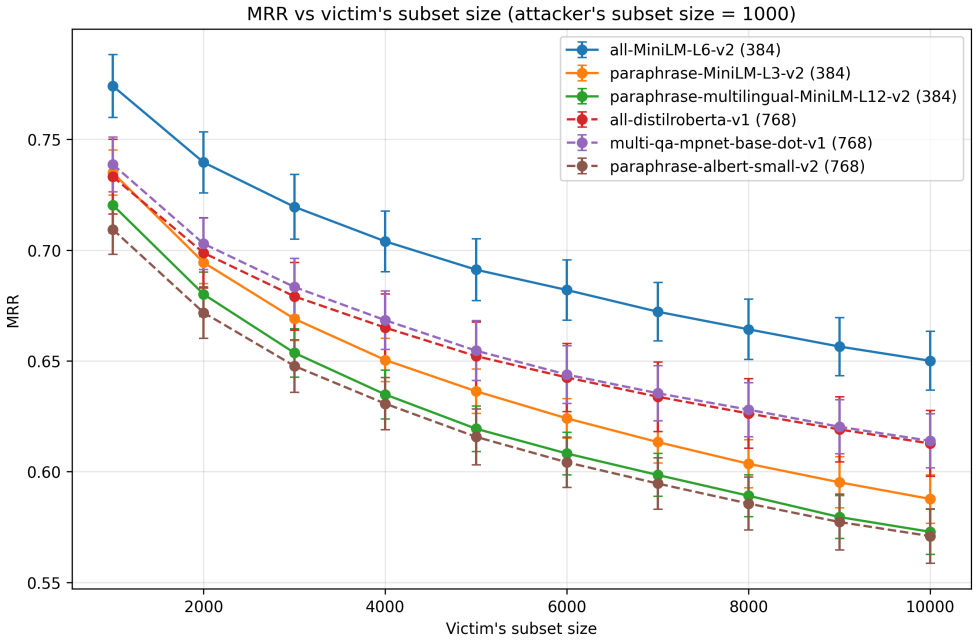

Figure 4.6 MRR for MovieLens item identification with Victim: Description / Attacker: Generated Description, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

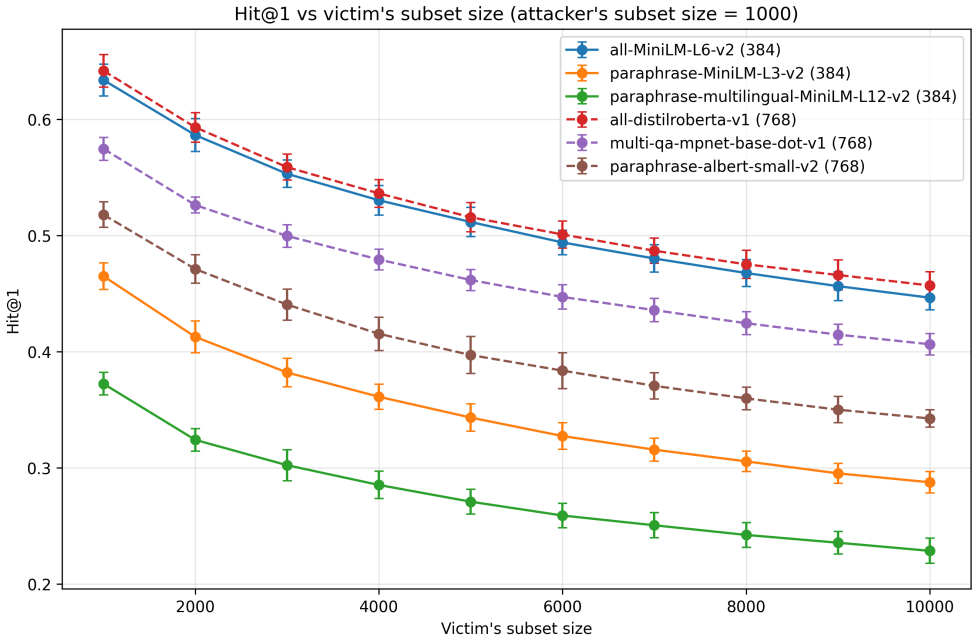

Figure 4.7 Hit@1 for Goodreads Books item identification with Victim: Description / Attacker: Generated Description, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

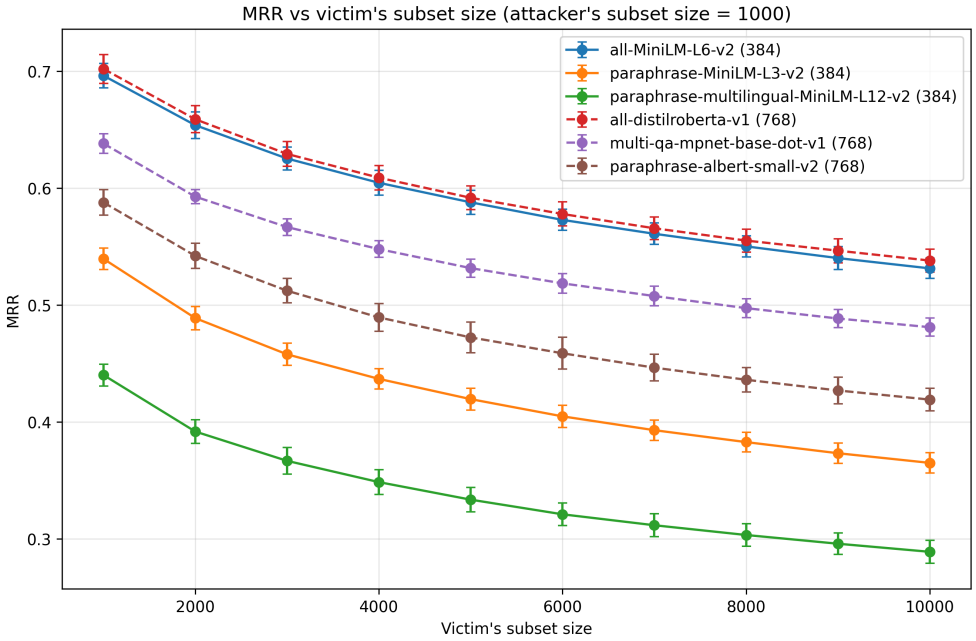

Figure 4.8 MRR for Goodreads Books item identification with Victim: Description / Attacker: Generated Description, fixed $|\bar{\mathcal{I}}| = 1{,}000$, and varying $|\mathcal{I}|$. Error bars show variation across seeds.

Dataset	Victim text	Attacker text	Hit@1 / MRR
MovieLens	title: description	title	0.98 / 0.99
MovieLens	description	title	0.80 / 0.85
MovieLens	title: description	generated description	0.64 / 0.69
MovieLens	description	generated description	0.59 / 0.65
Goodreads Books	title: description	title	0.89 / 0.93
Goodreads Books	title: description	generated description	0.73 / 0.80
Goodreads Books	description	title	0.53 / 0.61
Goodreads Books	description	generated description	0.46 / 0.54

Table 4.4 Summary of known candidate set results at $|\mathcal{I}| = 10{,}000$ and $|\bar{\mathcal{I}}| = 1{,}000$. The values report the best-performing model for each configuration.

the difficulty of the attack is driven primarily by the size of the leaked victim embedding pool.

Although increasing the size of both the attacker and victim sets in the closed-set scenario, where $\mathcal{I} = \bar{\mathcal{I}}$, reduces both MRR and Hit@1, the decline is generally small. For example, in the movie setting where x_j contains `title: description` and $\bar{x}_j$ contains `generated description`, stronger models such as `all-MiniLM-L6-v2` show only a minor decrease, typically less than 0.005 for every additional 1,000 items. Weaker models decline more noticeably; for instance, `paraphrase-MiniLM-L3-v2` drops by about 0.01 for every additional 1,000 items.

The effect is more pronounced in the weakest-performing setting, where x_j is set to `description` and $\bar{x}_j$ is set to `generated description`. In this case, increasing the set size leads to a larger performance drop, reaching approximately 0.04 for the weakest models and 0.03 for the strongest models per additional 1,000 items. This suggests that the closed-set identification setting is relatively stable as the number of items increases. Although performance decreases with larger attacker and victim sets, the decline is gradual, indicating that the method remains robust even as the search space grows.

MovieLens Dataset. For the MovieLens dataset, performance is highest when title information is available on both sides of the attack. The strongest configuration uses `title: description` as the victim-side text x_j and `title` as the attacker-side text $\bar{x}_j$. As shown in Figures 4.1 and 4.2, the best models remain close to perfect performance even when $|\mathcal{I}| = 10{,}000$, reaching approximately 0.98 Hit@1 and 0.99 MRR. This shows that movie titles provide a highly discriminative signal, especially when they appear in both the victim-side and attacker-side texts.

Performance decreases when the title is removed from the victim-side input, that is, when x_j is set to `description`. However, the attack remains strong. In this setting, the best model still reaches approximately 0.80 Hit@1 and 0.85

MRR at $|\mathcal{I}| = 10{,}000$. This suggests that movie titles and descriptions are strongly aligned in the embedding space. However, description-only settings should not be interpreted as completely title-free, since some descriptions may explicitly mention the movie title. For example, a description of *Toy Story* may begin with *"Toy Story is about . . ."*.

The attack becomes weaker when the attacker uses generated descriptions as $\bar{x}_j$, regardless of the victim-side text. As shown in Table 4.4, the two generated-description settings reach between 0.59 and 0.64 Hit@1 and between 0.65 and 0.69 MRR for the best-performing models when $|\bar{\mathcal{I}}| = 1{,}000$ and $|\mathcal{I}| = 10{,}000$. Even in the weakest text combination, shown in Figures 4.5 and 4.6, the results remain well above random retrieval. This indicates that generated descriptions preserve useful semantic information, but they are less reliable for exact item identification than titles.

Goodreads Books Dataset. For the Goodreads Books dataset, the same broad pattern holds, though absolute performance is lower across several settings. Performance is highest when title information is available on both sides of the attack. The strongest configuration uses `title: description` as the victim-side text x_j and `title` as the attacker-side text $\bar{x}_j$. As shown in Figures 4.3 and 4.4, the best model reaches approximately 0.89 Hit@1 and 0.93 MRR at $|\mathcal{I}| = 10{,}000$. This confirms that book titles are strong identifiers when present in the victim-side representation, though less so than in the corresponding movie setting.

Performance decreases when the attacker uses generated descriptions as $\bar{x}_j$, while the victim-side text still contains both title and description. In this setting, where x_j is set to `title: description`, the best models reach approximately 0.73 Hit@1 and 0.80 MRR at $|\mathcal{I}| = 10{,}000$. This shows that book descriptions contain substantial identifying information, but they are less reliable for exact item identification than titles.

The attack becomes considerably weaker when the title is removed from the victim-side input. When x_j is set to `description` and the attacker uses `title` as $\bar{x}_j$, the best model reaches approximately 0.53 Hit@1 and 0.61 MRR at $|\mathcal{I}| = 10{,}000$. In the fully description-based setting, where both the victim-side and attacker-side texts are description-based, the strongest models remain below 0.50 Hit@1, reaching approximately 0.46 Hit@1 and 0.54 MRR. As shown in Figures 4.7 and 4.8, these results indicate that removing the title from the victim-side representation substantially reduces book identifiability.

Comparison Between Datasets. Apart from differences in overall performance, the MovieLens and Goodreads Books datasets follow a similar pattern across text combinations, as shown in Table 4.4. In both datasets, the attack works best when title information is available on both the victim's and the attacker's sides. It becomes weaker when the match relies solely on descriptions. However, movie items are generally easier to identify than book items.

In the title-overlap setting, the best movie model achieves nearly 0.98 Hit@1 at $m = 10{,}000$, while the best book model reaches approximately 0.89.

This gap is larger when the victim embedding consists solely of descriptions, and the attacker uses titles. For movies, the best model reaches approximately 0.80 Hit@1 at $m = 10{,}000$. For Goodreads Books, the corresponding value is only about 0.53. This suggests that movie titles are more closely aligned with their descriptions than book titles are, at least for the embedding models evaluated here. A possible reason is that movie plot descriptions may more often repeat or directly mention the title, whereas book descriptions tend to be more varied and less directly tied to the title.

Description-based settings are weaker overall, especially for books. In the movie dataset, generated descriptions still allow non-trivial recovery: the best model achieves approximately 0.59 Hit@1 when both sides are description-based. In the corresponding book setting, the best models remain below 0.50 Hit@1. Overall, these results show that description-only item identification is possible, but it is much less reliable than title-based matching.

Defense mechanisms As explored, even in the worst-performing scenario, where both the attacker and the victim use descriptions, the attack still matches the corresponding items relatively successfully. This is likely due to common item-specific factors present in the text. For example, both generated and original descriptions may include titles, characters, themes, plot details, author-related information, or other distinctive cues that help identify individual items. We further investigate whether generated descriptions with reduced identifying information can serve as a defence mechanism. To do this, we generate descriptions using `gpt-oss-20b`, asking the model to produce factually correct descriptions while avoiding sensitive or directly identifying information about the items. These protected descriptions are then used as the victim-side text x_j.

For the MovieLens dataset, removing explicit identifying information from the victim-side description substantially reduces the attack's effectiveness, especially when the attacker uses titles. In this setting, the best model achieves only about 0.15 Hit@1 and 0.22 MRR in the closed-set scenario, where both the victim and attacker subsets have size 1,000. As the victim subset size increases, performance continues to decline, reaching approximately 0.08 Hit@1 and 0.13 MRR at $|\mathcal{I}| = 5{,}000$. This suggests that protected descriptions make title-based matching much more difficult, since the embeddings no longer contain the direct title-related cues that were highly useful in the previous experiments.

The same defensive effect is observed in the Goodreads Books dataset, although the attack remains stronger than in movies. When the attacker uses titles, the best model achieves 0.44 Hit@1 and 0.52 MRR at $|\bar{\mathcal{I}}| = |\mathcal{I}| = 1{,}000$. As the victim subset size increases to $|\mathcal{I}| = 5{,}000$, performance decreases to around 0.33 Hit@1 and 0.40 MRR for the best performing model. This

shows that removing explicit identifiers also reduces book identifiability, but protected book descriptions still retain more item-specific semantic information than protected movie descriptions.

However, the attack remains more effective when the attacker also uses descriptions. For movies, in the description-to-description setting, the best model reaches approximately 0.60 Hit@1 and 0.67 MRR at $|\bar{\mathcal{I}}| = |\mathcal{I}| = 1{,}000$. Even at $|\mathcal{I}| = 5{,}000$, with $|\bar{\mathcal{I}}|$ fixed at 1,000, the best model still obtains around 0.47 Hit@1 and 0.55 MRR. For books, the corresponding description-to-description results are slightly lower but still substantial: the best model reaches approximately 0.58 Hit@1 and 0.65 MRR at $|\mathcal{I}| = 1{,}000$, decreasing to around 0.46 Hit@1 and 0.53 MRR at $|\mathcal{I}| = 5{,}000$. This indicates that although protected descriptions remove obvious identifiers, they still retain enough semantic information to correctly match many items.

These results suggest that hiding titles and other explicit details reduces identifiability, but does not fully prevent recovery when the attacker can produce semantically similar descriptions. Since both sets of descriptions are generated by the same model, they may share similar wording, structure, or semantic emphasis, making corresponding items easier to identify even when explicit identifiers are removed. If the attacker used a different generation model, the overlap between the victim's and the attacker's descriptions might be smaller, reducing attack performance. However, since the attacker may also use the same or a similar model, these results represent a conservative setting.

Key Takeaways

Overall, the results from the closed and partially open candidate set demonstrate that similarity-based item identification can recover a substantial fraction of item identities from leaked embeddings. The attack is strongest when titles are available on both the victim and attacker sides, but remains effective under weaker assumptions through semantic alignment between titles and descriptions. Description-only representations reduce performance, but still often allow partial recovery.

These findings show that embedding-based data sharing can reveal meaningful information about the underlying items even when the original raw text is unavailable. The strength of the attack depends on the textual fields used to construct the embeddings, the dataset domain, and the embedding model used.

4.5.3 Item Identification Results with an Open Candidate Set

In this section, we evaluate the open candidate set scenario introduced in Section 3.3 during which the attacker does not possess the victim provider's item set but instead utilises an external candidate catalogue.

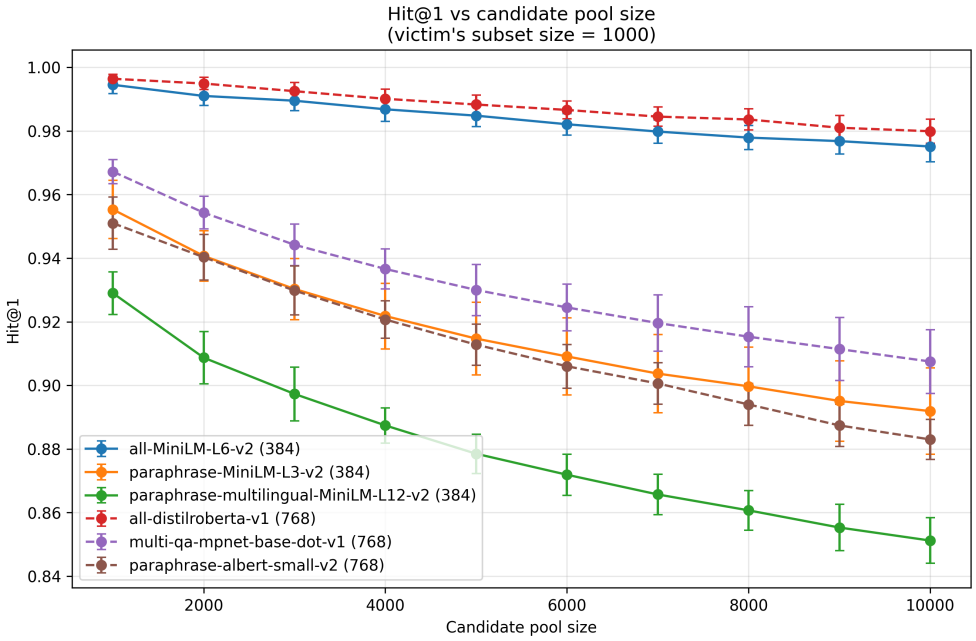

Figure 4.9 Hit@1 for MovieLens item identification with Victim: Title + Description / Attacker: Title, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

Similarly to the closed/partial candidate scenario, we vary both the victim-side text x_j and the attacker-side text $\bar{x}_j$. For the victim-side text, we again consider two formats: (1) the concatenation of title and description, and (2) the description alone. For the attacker-side text, we consider: (1) the item title, and (2) a generated description. Each explored combination is repeated across 10 random seeds for various sizes of $|\mathcal{I}|$ and $|\bar{\mathcal{I}}|$ to assess the stability of this approach.

The evaluated configurations are summarised in Table 4.5, the main results at $|\mathcal{I}| = 1{,}000, |\bar{\mathcal{I}}| = 10{,}000$. The full model-wise curves for the best embedding combinations are shown in Figures 4.9 and 4.10 for the MovieLens dataset and Figures 4.11 and 4.12 for the Goodreads Books dataset. Results of the worst performing scenario are shown in Figures 4.13 and 4.14 for MovieLens and Figures 4.15 and 4.16 for Goodreads Books.

Figure 4.9 shows the Hit@1 performance in the open candidate set scenario as the size of the external candidate pool increases. In this experiment, the victim subset size is fixed at 1,000 items, while the attacker's candidate catalogue varies from 1,000 to 10,000 items. Thus, the graph measures how well the attacker can identify the victim items when they search over increasingly large external candidate sets.

Although this plot is visually similar to the plots used for the closed and partial candidate set scenarios, it represents a different experimental setting.

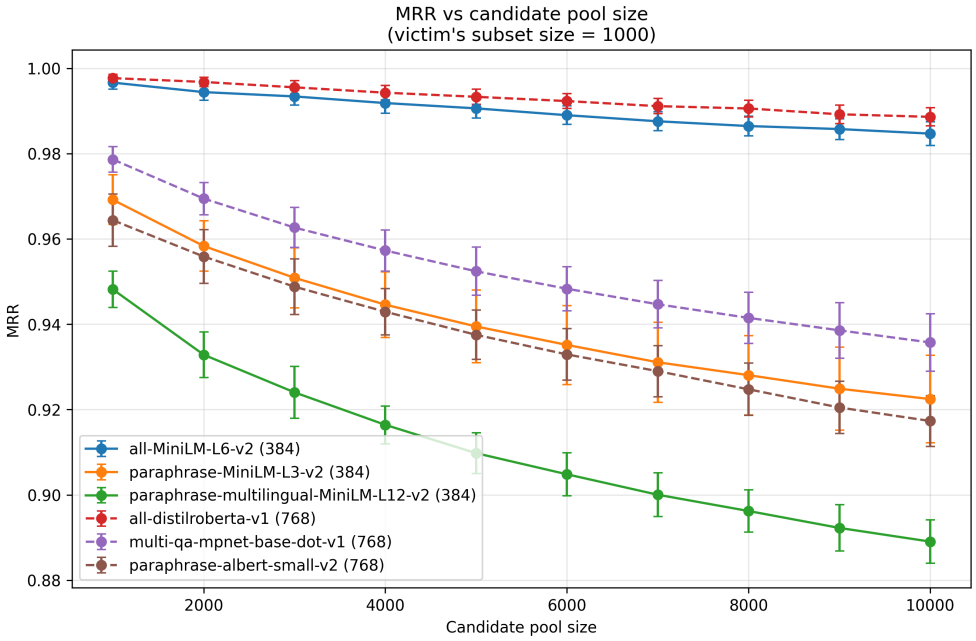

Figure 4.10 MRR for MovieLens item identification with Victim: Title + Description / Attacker: Title, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

In the closed and partial candidate set experiments, the x-axis corresponds to varying victim subset sizes, meaning the number of victim items attacked changes. In contrast, in the open candidate set scenario, the victim subset size remains fixed, and the x-axis instead represents the size of the attacker's external candidate pool. Therefore, the decrease in Hit@1 reflects the increasing difficulty of selecting the correct item from a larger candidate catalogue, rather than the effect of attacking a larger victim set.

Overall, Hit@1 decreases as the candidate pool grows, indicating that item identification becomes more difficult when the attacker must search over more possible candidates. However, the magnitude of this decrease differs across models. The strongest performance is observed for `all-distilroberta-v1` and `all-MiniLM-L6-v2`, which remain close to 0.98–1.00 even for the largest candidate pools. Other models show a more substantial decline, particularly `paraphrase-multilingual-MiniLM-L12-v2`, whose Hit@1 drops from approximately 0.93 to around 0.85 as the candidate pool increases. This suggests that the robustness of item identification in the open-candidate-set setting depends strongly on the embedding model used.

General Trends. Across all configurations and both datasets, increasing the candidate pool size $|\bar{\mathcal{I}}|$ consistently decreases both Hit@1 and MRR. This confirms that the primary difficulty in the open-candidate setting lies in distin-

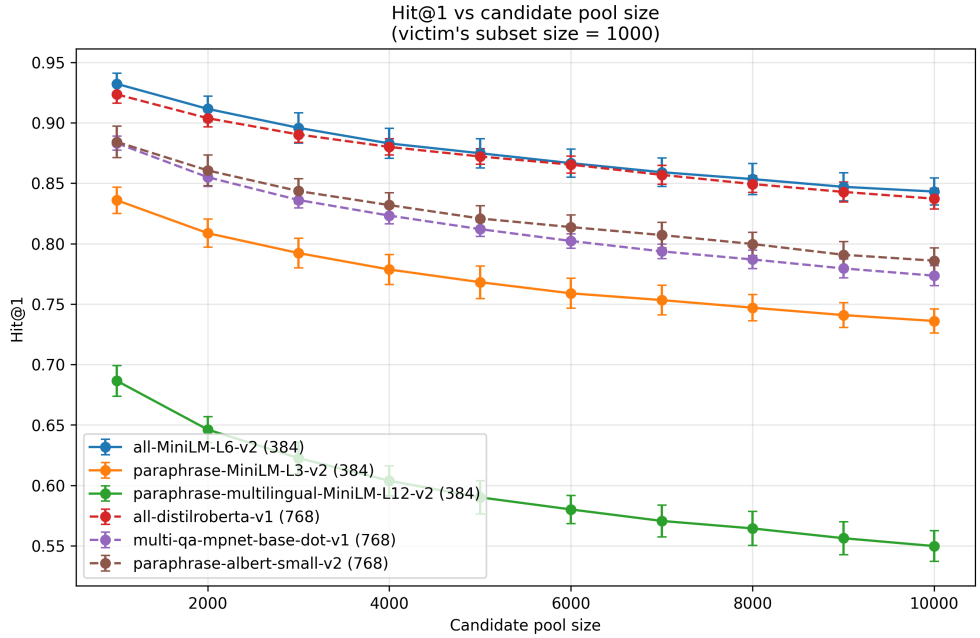

Figure 4.11 Hit@1 for Goodreads Books item identification with Victim: Title + Description / Attacker: Title, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

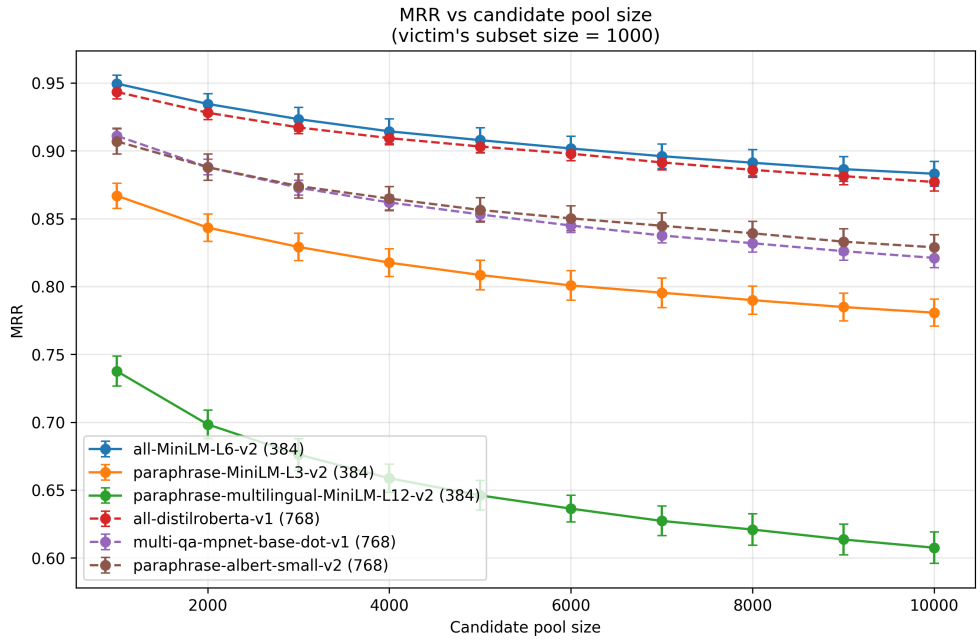

Figure 4.12 MRR for Goodreads Books item identification with Victim: Title + Description / Attacker: Title, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

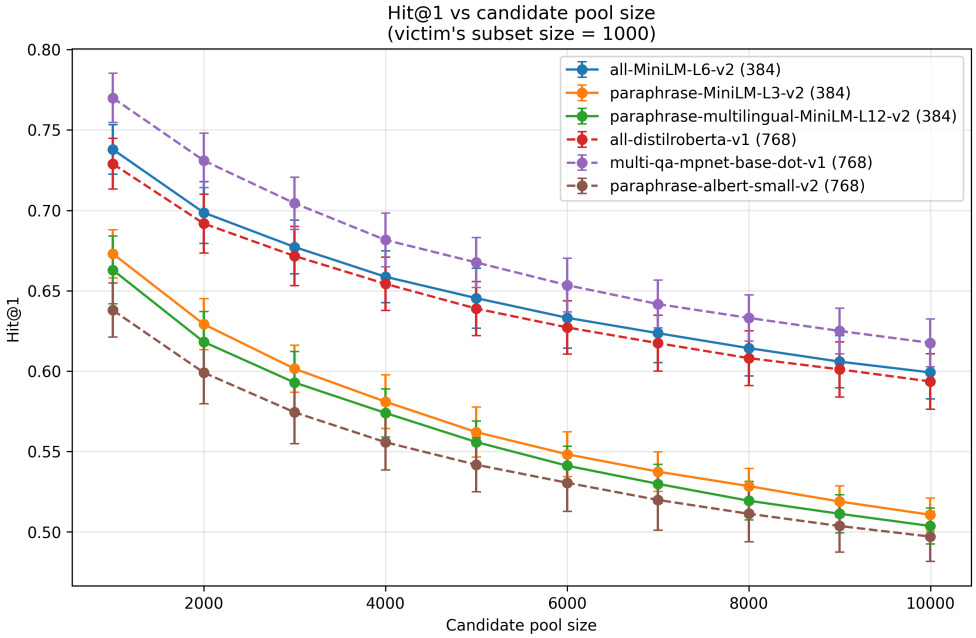

Figure 4.13 Hit@1 for MovieLens item identification with Victim: Description / Attacker: Generated Description, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

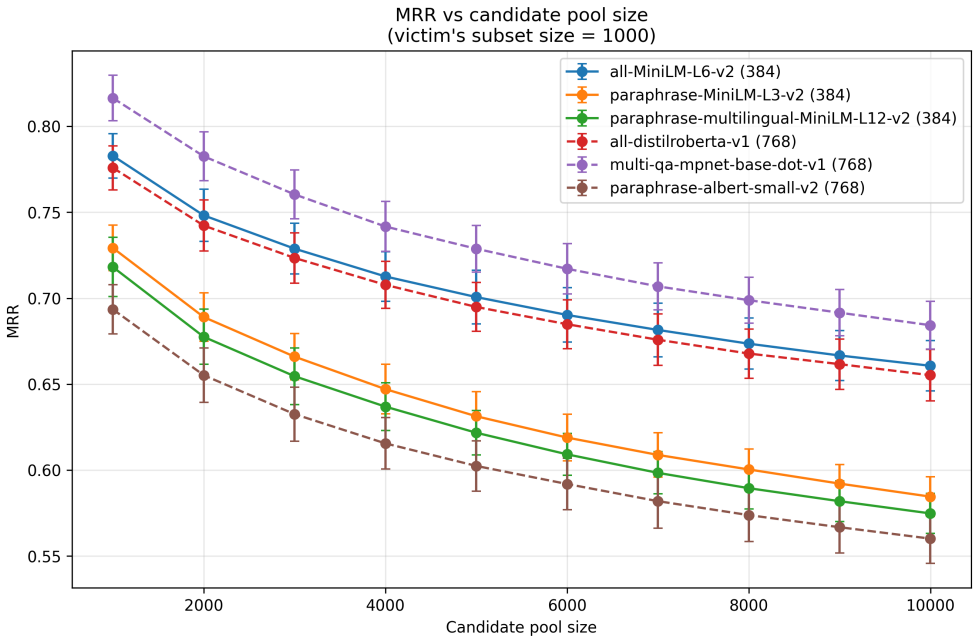

Figure 4.14 MRR for MovieLens item identification with Victim: Description / Attacker: Generated Description, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

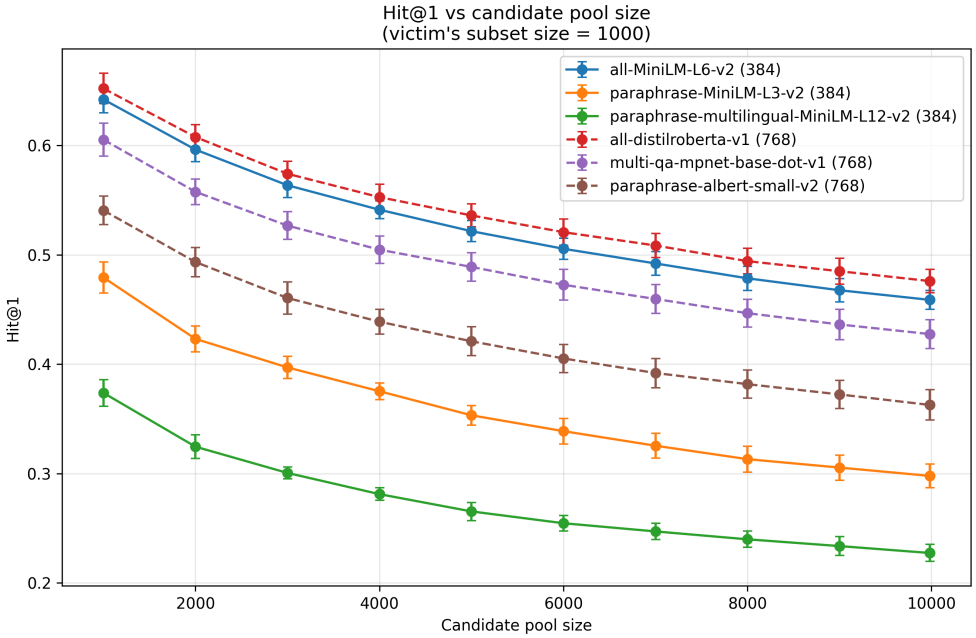

Figure 4.15 Hit@1 for Goodreads Books item identification with Victim: Description / Attacker: Generated Description, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

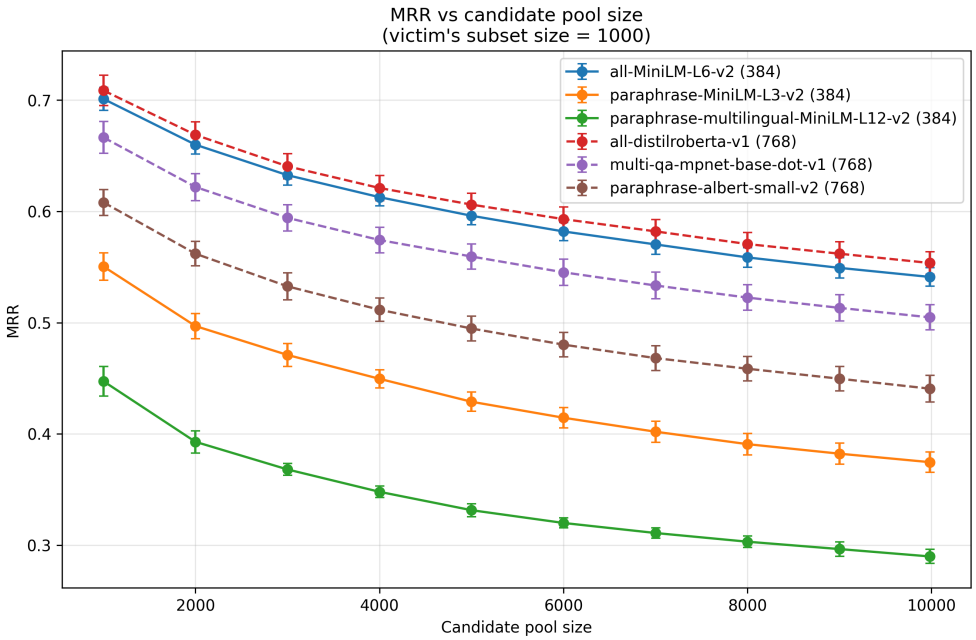

Figure 4.16 MRR for Goodreads Books item identification with Victim: Description / Attacker: Generated Description, fixed $|\mathcal{I}| = 1{,}000$, and varying $|\bar{\mathcal{I}}|$. Error bars show variation across seeds.

Dataset	Victim text	Attacker text	Best Hit@1 / MRR
MovieLens	title: description	title	0.98 / 0.99
MovieLens	description	title	0.80 / 0.85
MovieLens	title: description	generated description	0.63 / 0.68
MovieLens	description	generated description	0.62 / 0.68
Goodreads Books	title: description	title	0.84 / 0.88
Goodreads Books	title: description	generated description	0.76 / 0.82
Goodreads Books	description	title	0.48 / 0.55
Goodreads Books	description	generated description	0.47 / 0.55

Table 4.5 Summary of open candidate set results at $|\bar{\mathcal{I}}| = 10{,}000$ and $|\mathcal{I}| = 1{,}000$. Values correspond to the best-performing model in each configuration.

guishing the correct item from an increasing number of distractors. However, the decline is gradual rather than abrupt, indicating that embedding representations preserve sufficient structure for meaningful retrieval, even in large candidate spaces.

By contrast, varying the victim subset size $|\mathcal{I}|$ has only a limited effect on both Hit@1 and MRR across the two datasets, even when $|\mathcal{I}|$ increases from 1,000 to 10,000. This indicates that the difficulty of the attack is driven primarily by the size of the external candidate catalogue available to the attacker, rather than by the number of victim items being attacked.

This distinction is important because, in the open-candidate scenario, it is unlikely that the attacker has access to a candidate pool that exactly matches the victim provider's item set, i.e., $\mathcal{I} = \bar{\mathcal{I}}$. More realistic cases, therefore, involve a mismatch between the victim and attacker item sets, particularly settings where the attacker searches over a substantially larger external catalogue, such as $|\mathcal{I}| = 1{,}000$ and $|\bar{\mathcal{I}}| = 10{,}000$.

Similarly to the previous approach, in all scenarios, MRR remains higher than Hit@1, suggesting that even when exact top-1 identification fails, the correct item is often ranked among the top candidates.

MovieLens Dataset. For the MovieLens dataset, performance is highest when title information is available on both sides of the attack. The strongest configuration uses `title: description` as the victim-side text and `title` as the attacker-side text. In this setting, the best-performing models achieve near-perfect performance for small candidate pools and remain highly robust even when the attacker candidate pool increases to $|\bar{\mathcal{I}}| = 10{,}000$, with Hit@1 values close to 0.98. This indicates that movie titles provide a highly discriminative signal, particularly when title information is also present in the victim-side embedding.

Performance decreases when the title is removed from the victim-side input, that is, when the victim-side text is set to `description` while the attacker-side text remains `title`. Nevertheless, the attack stays effective: the best-

performing models still achieve approximately 0.80 Hit@1 at $|\bar{\mathcal{I}}| = 10{,}000$. This suggests a strong semantic alignment between movie titles and descriptions in the embedding space. However, description-only settings should not necessarily be interpreted as fully title-free, since some descriptions may explicitly contain the movie title or highly distinctive references to it, as previously explained.

By contrast, configurations that rely on generated descriptions on the attacker side are substantially weaker. This applies both when the victim-side text is `title: description` and when it is `description`. In both cases, using a generated description as the attacker-side text leads to a clear performance drop, with Hit@1 values of approximately 0.62–0.63. A similar level of performance is observed when both sides rely on description-based inputs. These results indicate that generated descriptions preserve useful semantic information but provide a weaker, less discriminative signal for exact item identification than titles.

Goodreads Books Dataset. The same overall trends are observed for the book dataset, although the absolute performance is lower than for movies. The strongest configuration is again the title-overlap setting, where the victim-side text is `title: description` and the attacker-side text is `title`. In this setting, the best-performing models achieve approximately 0.84 Hit@1 when the attacker candidate pool increases to $|\bar{\mathcal{I}}| = 10{,}000$. Although this still indicates strong item identifiability, the result is noticeably lower than in the MovieLens dataset, suggesting that book titles are less distinctive or less uniquely aligned with their descriptions.

When the attacker uses generated descriptions while the victim-side text remains `title: description`, performance also remains relatively strong, reaching approximately 0.76 Hit@1. This suggests that book descriptions contain rich semantic signals that can support item identification even without direct title matching on the attacker's side.

More challenging configurations lead to a larger decrease in performance. When the title is removed from the victim-side text and the victim uses only `description`, performance drops substantially, both when the attacker-side text is `title` and when it is a `generated description`. In these settings, Hit@1 decreases to approximately 0.47–0.48. This indicates that, for Goodreads Books, removing title information from the victim-side representation significantly reduces identifiability. Compared with the MovieLens dataset, the weaker description-only performance suggests that book descriptions may be less directly aligned with titles, more semantically broad, or less distinctive at the individual-item level.

Cross-Dataset Comparison. Comparing the two datasets reveals clear domain-specific differences. In most configurations, MovieLens achieves higher performance than Goodreads Books, particularly when the attacker uses `title`

as the attacker-side text. For example, in the strongest title-overlap setting, where the victim-side text is `title: description` and the attacker-side text is `title`, MovieLens reaches 0.98 Hit@1 and 0.99 MRR, compared with 0.84 Hit@1 and 0.88 MRR for Goodreads Books. This suggests that movie titles are more distinctive and more strongly aligned with their descriptions in the embedding space.

The main exception occurs when the victim text is `title: description` and the attacker uses a generated description. In this configuration, Goodreads Books achieves higher performance, with 0.76 Hit@1 and 0.82 MRR, compared with 0.63 Hit@1 and 0.68 MRR for MovieLens. This suggests that generated descriptions may capture book-specific semantic content more effectively than movie-specific content, though the effect may also be influenced by descriptions that include title-specific information.

The weakest results are observed for Goodreads Books when the victim-side text is limited to `description`. In these configurations, performance drops to 0.48/0.55 when the attacker uses `title` and 0.47/0.55 when the attacker uses a generated description. Nevertheless, these scores remain well above random retrieval, indicating that the embeddings still leak meaningful information about the underlying items. Therefore, even in the weaker-performing dataset and configurations, the attacker can recover partial information about the items represented in the victim embeddings.

Defense mechanisms Similarly to the previous approach, we evaluate if generated descriptions with reduced item-identifying information can serve as a defence mechanism in the open-candidate-set scenario. Here, the victim-side text is replaced with descriptions generated by `gpt-oss-20b`, which is instructed to preserve factual correctness while avoiding explicit item-identifying details. The attacker then attempts to identify the corresponding items using either `title` or generated descriptions as the attacker-side text.

The results show that this defence substantially weakens the attack, especially when the attacker relies on titles. For MovieLens, the title-based attacker achieves only 0.11 Hit@1 and 0.17 MRR at $|\bar{\mathcal{I}}| = 1{,}000$ for the best-performing model. When the attacker candidate pool increases to $|\bar{\mathcal{I}}| = 10{,}000$, performance drops further to around 0.04 Hit@1 and 0.07 MRR. This is a large decrease compared with the earlier title-overlap setting, where MovieLens reached approximately 0.98 Hit@1 and 0.99 MRR at the same candidate pool size. This suggests that much of the previous identification success was driven by direct or indirect title-specific cues in the victim-side text.

For Goodreads Books, the same defensive effect is observed, although the title-based attacker remains more successful than in the MovieLens dataset. The best model reaches approximately 0.35 Hit@1 and 0.41 MRR at $|\bar{\mathcal{I}}| = 1{,}000$, decreasing to around 0.22 Hit@1 and 0.28 MRR at $|\bar{\mathcal{I}}| = 10{,}000$. This indicates that protected book descriptions still retain more item-specific se-

mantic information than protected movie descriptions, but the attack is nevertheless much weaker than in the original title-based configurations.

The attack stays more effective when the attacker also uses generated descriptions. For MovieLens, description-to-description matching reaches approximately 0.42 Hit@1 and 0.51 MRR at $|\bar{\mathcal{I}}| = 1{,}000$, and decreases to around 0.24 Hit@1 and 0.32 MRR at $|\bar{\mathcal{I}}| = 10{,}000$. For Goodreads Books, the corresponding values are slightly lower but still substantial: the best model achieves 0.43 Hit@1 and 0.50 MRR at $|\bar{\mathcal{I}}| = 1{,}000$, dropping to around 0.28 Hit@1 and 0.34 MRR at $|\bar{\mathcal{I}}| = 10{,}000$.

Overall, generated descriptions with reduced identifying information significantly reduce item identifiability, particularly against attackers using titles. However, the defence does not eliminate the risk. When the attacker also uses generated descriptions, both datasets still show non-trivial matching performance even at $|\bar{\mathcal{I}}| = 10{,}000$. This suggests that, even after explicit identifiers are removed, the embeddings continue to encode semantic information that can support item recovery. Therefore, protected descriptions should be interpreted as a mitigation strategy rather than a complete privacy-preserving solution.

Key Takeaways

Overall, the unknown candidate set experiments demonstrate that item identification from embeddings remains feasible even when the attacker must search over large and noisy candidate pools. Increasing the candidate pool size reduces performance, but does not eliminate the identifying signal.

The attack is strongest when explicit identifying information, such as titles, is present on both sides, especially when the better-performing embedding models are used. However, meaningful recovery remains possible through semantic alignment, even in the absence of direct overlap. These findings highlight that embedding-based data sharing can expose substantial information about the underlying items, even under realistic, constrained attack assumptions.

4.5.4 Effect of the Used Embedding Model

The preceding experiments show that item identification performance depends strongly on the embedding model, but the ordering of models varies across scenarios. In the original title and description-based experiments, strong performance is often achieved by `all-MiniLM-L6-v2`, `all-distilroberta-v1`, and `multi-qa-mpnet-base-dot-v1`. However, in defence experiments using protected descriptions, `multi-qa-mpnet-base-dot-v1` becomes the strongest model across several configurations, particularly when the victim-side text is a protected description.

This suggests that model effectiveness depends not only on the embedding architecture, but also on the type of textual signal available to the attacker. When explicit identifiers such as titles are present, several models can exploit this direct overlap effectively. Once these identifiers are removed, the attack relies more heavily on broader semantic similarity, where models such as `multi-qa-mpnet-base-dot-v1` appear to preserve more useful matching information.

The observed differences are not explained solely by embedding dimensionality. Both 384-dimensional and 768-dimensional models appear among the stronger and weaker configurations, indicating that larger embeddings do not necessarily lead to greater item identifiability. Instead, performance is likely influenced by a combination of training data, training objective, architecture, and input length constraints. For example, some weaker-performing models, such as `paraphrase-MiniLM-L3-v2` and `paraphrase-albert-small-v2`, tend to perform less consistently across the evaluated settings, while models trained on broader data often provide more discriminative representations for item matching.

4.5.5 Recommendation

A language model used to embed item information should not be chosen solely for its perceived privacy-preserving properties, as the resulting embeddings must also be effective for the downstream tasks they support. To examine the trade-off between privacy and recommendation performance, we introduce a toy model to evaluate rating-prediction quality in an explicit feedback setting.

The recommendation experiment follows a matrix-factorisation-style approach for explicit feedback. Given the rating matrix R, ratings are approximated as

$$R \approx UV^\top,$$

where U denotes learned user embeddings and V denotes item embeddings.

In contrast to standard matrix factorisation, V is fixed and corresponds to the LLM-generated item embeddings for a given text representation and embedding model. Only the user embeddings U are regularised to reduce overfitting. Performance is evaluated on held-out ratings using RMSE, where lower values indicate better recommendation accuracy.

Table 4.6 reports recommendation performance on the Goodreads Books dataset using RMSE, where lower values indicate better predictive accuracy. The results show that both the textual input and the embedding model affect recommendation quality.

Across all text representations, `multi-qa-mpnet-base-dot-v1` achieves the lowest RMSE, with values close to 1.00 for `title: description`, `original description`, `generated descriptions`, and `protected descriptions`. This indicates that this model produces the most useful item representations for the recommendation task among the evaluated embedding models. Using only

Table 4.6 Recommendation performance on the Goodreads Books dataset for different textual inputs and embedding models. Values report RMSE; lower values indicate better performance.

Text representation	multi-qa	MiniLM-L6	distilroberta	MiniLM-L3	multilingual-MiniLM	albert-small
title: description	0.995	1.643	1.706	1.447	1.318	1.371
title	1.072	1.650	1.648	2.056	1.453	1.496
generated description	1.003	1.506	1.502	1.400	1.327	1.328
description	0.996	1.600	1.620	1.449	1.348	1.366
protected description	0.997	1.376	1.446	1.328	1.307	1.293

`title` leads to a slightly higher RMSE of 1.072, suggesting that description-based information provides additional useful semantic signal.

The impact of the text representation differs across models. For several models, protected descriptions achieve lower RMSE than the original `title: description` representation. For example, `paraphrase-albert-small-v2` improves from 1.371 to 1.293, and `paraphrase-MiniLM-L3-v2` improves from 1.447 to 1.328. A similar improvement is observed for `all-MiniLM-L6-v2`, where RMSE decreases from 1.643 to 1.376. This suggests that removing explicit identifying details does not necessarily degrade recommendation performance and may even yield cleaner, more general semantic representations for some models.

Overall, these results indicate that privacy-oriented text transformation can reduce item identification while preserving recommendation utility. However, this effect is model-dependent. `multi-qa-mpnet-base-dot-v1` remains clearly strongest across all settings, while the other models are more sensitive to the type of textual input used to construct the embeddings.

It is also important to note that the model ranking for recommendation performance does not exactly match the ranking observed in the item identification experiments. In the item identification setting, the `paraphrase-*` models were often among the weaker models, while in several cases performing well on the recommendation task. `paraphrase-multilingual-MiniLM-L12-v2` and `paraphrase-albert-small-v2` achieve competitive RMSE values, especially when protected generated descriptions are used.

By contrast, `multi-qa-mpnet-base-dot-v1` performs well in both tasks: it achieves the lowest RMSE in the recommendation experiments, while also being among the strongest models for item identification. This suggests that highly informative embedding spaces may improve recommendation quality, but may also increase the risk of item identifiability.

These results highlight a trade-off between privacy and utility in the embedding model selection process. Choosing a sentence transformer primarily to mitigate item inversion risk is insufficient on its own, as the embeddings must also remain useful for the recommendation task they are intended to support. Therefore, embedding model selection should be guided not only by privacy considerations but also by recommendation performance, particularly in settings where embeddings are shared with external parties.

4.6 Security Implications and Defences

The results presented indicate that stored item embeddings should be treated as sensitive data. Embeddings may reveal information about the embedded items, the embedding model, or both. This is particularly relevant for systems that store embeddings in vector databases, share them with external services, or expose them through APIs.

The most immediate defence should be strict access control. Embeddings should only be exposed when necessary, and access to vector databases should be protected similarly to access to the underlying textual data. Authentication, logging, rate limiting, and tenant isolation can reduce the likelihood that an attacker obtains enough embeddings to perform a reliable attack.

A second mitigation is data minimisation at the input level. The experiments show that titles, detailed descriptions, and other item-specific cues can substantially increase identifiability. Using fewer identifying textual inputs, such as protected descriptions that avoid explicit titles or distinctive details, can reduce the effectiveness of attacks. However, this does not eliminate the risk, since semantic similarity alone may still support item matching.

Another defence is to reduce the direct comparability of released embeddings. Possible approaches include learned projections, secret rotations, dimensionality reduction, or other private transformations before storage or sharing. These methods may make it harder to compare leaked embeddings with embeddings generated from public models. However, they must be evaluated carefully, because transformations that preserve similarity for recommendation or retrieval may also preserve enough structure for inference.

Noise-based defences may also weaken item matching by perturbing the embedding space. This creates a privacy–utility trade-off: too much noise can harm recommendation, retrieval, or ranking quality, while too little noise may provide limited protection. The appropriate level of perturbation, therefore, depends on the application and should be selected empirically.

Model secrecy should not be considered sufficient protection. The model identification results indicate that an attacker may infer the embedding model from leaked embeddings within a realistic candidate set. Hiding the model choice or deployment configuration may increase difficulty, but it does not provide a reliable defence on its own.

These findings highlight the need to consider embeddings as part of the system's security boundary. Embeddings remain useful for recommendation and retrieval, but their storage and sharing should be governed by explicit security assumptions. Effective protection requires combining access control, input data minimisation, and empirical evaluation of embedding transformations or perturbations.

4.7 Limitations

Several limitations should be considered when interpreting the results. First, the evaluation is limited to two domains: movies and books. Although similar trends are observed across both datasets, absolute performance differs, suggesting that domain-specific properties, such as description richness, distinctiveness, and consistency, influence item identifiability. Therefore, the results should not be interpreted as domain-independent without validation on additional datasets.

Second, the experiments focus on item embeddings generated from textual metadata. In real recommender systems, embeddings may also include structured metadata, interaction histories, collaborative signals, user-specific features, or multimodal information. These additional signals could either increase identifiability by making items more distinctive or reduce reproducibility by pushing the embeddings farther from what an attacker can generate from public text alone.

The experiments are also restricted to English-language items. This removes the need for the attacker to infer the language or localisation of the original input. In practice, platforms may store items in multiple languages or use different titles and descriptions across markets. Language uncertainty could therefore affect item matching and model identification.

Another limitation concerns embedding dimensionality. In the model identification experiments, the dimensionality of the victim embeddings is assumed to be known, allowing incompatible candidate models to be excluded. This simplifies the attack. If the victim applies post-processing, such as principal component analysis or other dimensionality reduction, the observed dimension may no longer correspond to the model's native output size, making model identification more difficult.

The attacker model also assumes that the attacker knows the victim domain and can construct a meaningful candidate pool. This is plausible for public domains such as movies and books, where metadata is widely available. However, it may be less realistic for specialised or proprietary domains where the item catalogue is difficult to enumerate. In such cases, item identification may become less reliable, although model identification could still be possible using related auxiliary data.

Chapter 5

Conclusion

Large language models have become an important component of modern recommender systems, especially through their ability to generate semantic representations of users and items. Although these embeddings are often treated as privacy-preserving because they are high-dimensional and not directly human-readable, this thesis shows that this assumption should be treated with caution. Even without reconstructing the original text, embeddings can preserve enough semantic and structural information to support item-level inference.

The main goal of this work was to investigate privacy risks associated with sharing LLM-generated item embeddings in recommender system settings. Instead of focusing on direct embedding inversion, we considered a more constrained setting in which an attacker attempts to identify the embedding model and then infer which items correspond to the observed embeddings. This reframes the privacy risk from exact text reconstruction to item-level identifiability.

All proposed tasks from the introduction were addressed. The thesis reviewed related work on LLM-based recommender systems, embedding representations, information leakage, and embedding inversion. It then formalised item-identifiability scenarios based on the attacker's access to item sets, identifiers, and embedding models. These scenarios were implemented and evaluated on two public datasets, MovieLens and Goodreads Books, followed by a discussion of their privacy and security implications.

The experiments show that LLM-generated embeddings can reveal meaningful information about the underlying items, especially when the attacker has access to candidate metadata or can approximate the embedding-generation process. Similarity-based matching is often sufficient to recover corresponding or closely related items. Identification is strongest when title information is present, but even description-based settings remain vulnerable because descriptions may contain item-specific cues or preserve distinctive semantic information.

Beyond the original proposed tasks, this thesis also explored several additional aspects. We tested a simple defence mechanism using generated descriptions that avoid explicit identifying information. This reduced attack success, particularly against title-based attackers, but did not fully remove the risk when the attacker used semantically similar descriptions.

We additionally ran a basic recommendation experiment to examine how the same embedding choices affect both recommendation utility and item identifiability. Using a matrix-factorisation-style setup, we found that privacy-oriented text transformations do not necessarily destroy recommendation quality. The results also suggest that the model ranking for recommendation is not identical to the ranking for item identification, highlighting a practical privacy–utility trade-off.

Overall, this thesis shows that embedding privacy in recommender systems should not be limited to text reconstruction. Item-level identifiability is itself a relevant privacy and security risk. Stored or shared embeddings should therefore be treated as sensitive data, particularly when the underlying item catalogue is public, partially known, or easy to approximate.

Future work could extend this study to additional domains, datasets, embedding models, and model families. It should also consider user embeddings, sequential interaction histories, and hybrid user-item representations, where privacy risks may be more sensitive.

In conclusion, LLM-generated embeddings provide clear benefits for recommender systems, but they should not be assumed to hide the information from which they were derived. Even without direct text reconstruction, an attacker may infer item identities, semantic content, or the generating embedding model from the embedding space. Secure embedding-based data sharing, therefore, requires explicit threat modelling, privacy evaluation, and appropriate safeguards.

Bibliography

1. NAVEED, Humza; KHAN, Asad Ullah; QIU, Shi; SAQIB, Muhammad; ANWAR, Saeed; USMAN, Muhammad; AKHTAR, Naveed; BARNES, Nick; MIAN, Ajmal. A Comprehensive Overview of Large Language Models. *ACM Trans. Intell. Syst. Technol.* 2025, vol. 16, no. 5. ISSN 2157-6904. Available from DOI: 10.1145/3744746.
2. LIN, Jianghao; DAI, Xinyi; XI, Yunjia; LIU, Weiwen; CHEN, Bo; ZHANG, Hao; LIU, Yong; WU, Chuhan; LI, Xiangyang; ZHU, Chenxu; GUO, Huifeng; YU, Yong; TANG, Ruiming; ZHANG, Weinan. How Can Recommender Systems Benefit from Large Language Models: A Survey. *ACM Trans. Inf. Syst.* 2025, vol. 43, no. 2. ISSN 1046-8188. Available from DOI: 10.1145/3678004.
3. MORRIS, John; KULESHOV, Volodymyr; SHMATIKOV, Vitaly; RUSH, Alexander. Text Embeddings Reveal (Almost) As Much As Text. In: BOUAMOR, Houda; PINO, Juan; BALI, Kalika (eds.). *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, 2023, pp. 12448–12460. Available from DOI: 10.18653/v1/2023.emnlp-main.765.
4. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: BURSTEIN, Jill; DORAN, Christy; SOLORIO, Thamar (eds.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. Available from DOI: 10.18653/v1/N19-1423.
5. SONG, Congzheng; RAGHUNATHAN, Ananth. Information Leakage in Embedding Models. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 377–390. CCS '20. ISBN 9781450370899. Available from DOI: 10.1145/3372297.3417270.

6. KOJIMA, Takeshi; GU, Shixiang Shane; REID, Machel; MATSUO, Yutaka; IWASAWA, Yusuke. Large language models are zero-shot reasoners. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. New Orleans, LA, USA: Curran Associates Inc., 2022. NIPS '22. ISBN 9781713871088.

7. REIMERS, Nils; GUREVYCH, Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: INUI, Kentaro; JIANG, Jing; NG, Vincent; WAN, Xiaojun (eds.). *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3982–3992. Available from DOI: 10.18653/v1/D19-1410.

8. RAFFEL, Colin; SHAZEER, Noam; ROBERTS, Adam; LEE, Katherine; NARANG, Sharan; MATENA, Michael; ZHOU, Yanqi; LI, Wei; LIU, Peter J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 2020, vol. 21, no. 1. ISSN 1532-4435.

9. LIN, Ailiang; LI, Zhuoyun; FUNAKOSHI, Kotaro; OKUMURA, Manabu. Causal2Vec: Improving Decoder-only LLMs as Versatile Embedding Models. *arXiv preprint arXiv:2507.23386*. 2025.

10. LIU, Qi; KUSNER, Matt J; BLUNSOM, Phil. A survey on contextual embeddings. *arXiv preprint arXiv:2003.07278*. 2020.

11. KUDO, Taku; RICHARDSON, John. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In: BLANCO, Eduardo; LU, Wei (eds.). *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 66–71. Available from DOI: 10.18653/v1/D18-2012.

12. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 2013.

13. PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher. GloVe: Global Vectors for Word Representation. In: MOSCHITTI, Alessandro; PANG, Bo; DAELEMANS, Walter (eds.). *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. Available from DOI: 10.3115/v1/D14-1162.

14. *Transformers – Hugging Face — huggingface.co* [https://huggingface.co/docs/transformers/index]. [N.d.]. [Accessed 20-04-2026].

15. XI, Yunjia; LIU, Weiwen; LIN, Jianghao; CAI, Xiaoling; ZHU, Hong; ZHU, Jieming; CHEN, Bo; TANG, Ruiming; ZHANG, Weinan; YU, Yong. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. In: *Proceedings of the 18th ACM Conference on Recommender Systems*. Bari, Italy: Association for Computing Machinery, 2024, pp. 12–22. RecSys '24. ISBN 9798400705052. Available from DOI: 10.1145/3640457.3688104.

16. CARRANZA, Aldo; FARAHANI, Rezsa; PONOMAREVA, Natalia; KURAKIN, Alexey; JAGIELSKI, Matthew; NASR, Milad. Synthetic Query Generation for Privacy-Preserving Deep Retrieval Systems using Differentially Private Language Models. In: DUH, Kevin; GOMEZ, Helena; BETHARD, Steven (eds.). *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Mexico City, Mexico: Association for Computational Linguistics, 2024, pp. 3920–3930. Available from DOI: 10.18653/v1/2024.naacl-long.217.

17. WU, Chuhan; WU, Fangzhao; QI, Tao; HUANG, Yongfeng. Empowering News Recommendation with Pre-trained Language Models. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 1652–1656. SIGIR '21. ISBN 9781450380379. Available from DOI: 10.1145/3404835.3463069.

18. LI, Ruyu; DENG, Wenhao; CHENG, Yu; YUAN, Zheng; ZHANG, Jiaqi; YUAN, Fajie. Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights. In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025, pp. 1643–1653.

19. SCHEIN, Andrew I.; POPESCUL, Alexandrin; UNGAR, Lyle H.; PENNOCK, David M. Methods and metrics for cold-start recommendations. In: *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Tampere, Finland: Association for Computing Machinery, 2002, pp. 253–260. SIGIR '02. ISBN 1581135610. Available from DOI: 10.1145/564376.564421.

20. PARK, Yoon-Joo; TUZHILIN, Alexander. The long tail of recommender systems and how to leverage it. In: *Proceedings of the 2008 ACM Conference on Recommender Systems*. Lausanne, Switzerland: Association for Computing Machinery, 2008, pp. 11–18. RecSys '08. ISBN 9781605580937. Available from DOI: 10.1145/1454008.1454012.

21. LI, Xinhang; CHEN, Chong; ZHAO, Xiangyu; ZHANG, Yong; XING, Chunxiao. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443*. 2023.

22. HUA, Wenyue; GE, Yingqiang; XU, Shuyuan; JI, Jianchao; LI, Zelong; ZHANG, Yongfeng. UP5: Unbiased Foundation Model for Fairness-aware Recommendation. In: GRAHAM, Yvette; PURVER, Matthew (eds.). *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. St. Julian's, Malta: Association for Computational Linguistics, 2024, pp. 1899–1912. Available from DOI: 10.18653/v1/2024.eacl-long.114.

23. GENG, Shijie; LIU, Shuchang; FU, Zuohui; GE, Yingqiang; ZHANG, Yongfeng. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In: *Proceedings of the 16th ACM Conference on Recommender Systems*. Seattle, WA, USA: Association for Computing Machinery, 2022, pp. 299–315. RecSys '22. ISBN 9781450392785. Available from DOI: 10.1145/3523227.3546767.

24. YANG, Bowen; HAN, Cong; LI, Yu; ZUO, Lei; YU, Zhou. Improving Conversational Recommendation Systems' Quality with Context-Aware Item Meta-Information. In: CARPUAT, Marine; MARNEFFE, Marie-Catherine de; MEZA RUIZ, Ivan Vladimir (eds.). *Findings of the Association for Computational Linguistics: NAACL 2022*. Seattle, United States: Association for Computational Linguistics, 2022, pp. 38–48. Available from DOI: 10.18653/v1/2022.findings-naacl.4.

25. WANG, Xiaolei; TANG, Xinyu; ZHAO, Xin; WANG, Jingyuan; WEN, Ji-Rong. Rethinking the Evaluation for Conversational Recommendation in the Era of Large Language Models. In: BOUAMOR, Houda; PINO, Juan; BALI, Kalika (eds.). *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, 2023, pp. 10052–10065. Available from DOI: 10.18653/v1/2023.emnlp-main.621.

26. GAO, Yunfan; SHENG, Tao; XIANG, Youlin; XIONG, Yun; WANG, Haofen; ZHANG, Jiawei. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*. 2023.

27. HUANG, Yu-Hsiang; TSAI, Yuche; HSIAO, Hsiang; LIN, Hong-Yi; LIN, Shou-De. Transferable Embedding Inversion Attack: Uncovering Privacy Risks in Text Embeddings without Model Queries. In: KU, Lun-Wei; MARTINS, Andre; SRIKUMAR, Vivek (eds.). *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, 2024, pp. 4193–4205. Available from DOI: 10.18653/v1/2024.acl-long.230.

28. CHEN, Yiyi; LENT, Heather; BJERVA, Johannes. Text Embedding Inversion Security for Multilingual Language Models. In: KU, Lun-Wei; MARTINS, Andre; SRIKUMAR, Vivek (eds.). *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, 2024, pp. 7808–7827. Available from DOI: 10.18653/v1/2024.acl-long.422.

29. CHEN, Yiyi; XU, Qiongkai; BJERVA, Johannes. ALGEN: Few-shot Inversion Attacks on Textual Embeddings via Cross-Model Alignment and Generation. In: CHE, Wanxiang; NABENDE, Joyce; SHUTOVA, Ekaterina; PILEHVAR, Mohammad Taher (eds.). *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vienna, Austria: Association for Computational Linguistics, 2025, pp. 24330–24348. ISBN 979-8-89176-251-0. Available from DOI: 10.18653/v1/2025.acl-long.1185.

30. LIN, Chin-Yew. ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 74–81. Available also from: https://aclanthology.org/W04-1013/.

31. ZHANG, Jinghao; LIU, Yuting; LIU, Qiang; WU, Shu; GUO, Guibing; WANG, Liang. Stealthy Attack on Large Language Model based Recommendation. In: KU, Lun-Wei; MARTINS, Andre; SRIKUMAR, Vivek (eds.). *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, 2024, pp. 5839–5857. Available from DOI: 10.18653/v1/2024.acl-long.318.

32. WANG, Yubo; TANG, Min; SHEN, Nuo; CUI, Shujie; WANG, Weiqing. Privacy Risks of LLM-Empowered Recommender Systems: An Inversion Attack Perspective. In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 2025, pp. 812–821.

33. KASALICKÝ, Petr; SPIŠÁK, Martin; VANČURA, Vojtěch; BOHUNĚK, Daniel; ALVES, Rodrigo; KORDÍK, Pavel. The Future is Sparse: Embedding Compression for Scalable Retrieval in Recommender Systems. In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. Association for Computing Machinery, 2025, pp. 1099–1103. RecSys '25. ISBN 9798400713644. Available from DOI: 10.1145/3705328.3748147.

34. VANČURA, Vojtěch; SPIŠÁK, Martin; ALVES, Rodrigo; PEŠKA, Ladislav. Efficient Learning of Sparse Representations from Interactions. In: *Proceedings of the ACM Web Conference 2026*. United Arab Emirates: Association for Computing Machinery, 2026, pp. 8577–8580. WWW '26. ISBN 9798400723070. Available from DOI: 10.1145/3774904.3792914.

35. SPIŠÁK, Martin; ALVES, Rodrigo; KELLEHER, Thomas; SHEPPARD, Jason; FIEDLER, Ondřej; KOSOVRASTI, Ergys; VANČURA, Vojtěch; KASALICKÝ, Petr; KORDÍK, Pavel. Segment-Aware Analytics for Real-Time Editorial Support in Media Groups. 2025.

36. *Pretrained Models; Sentence Transformers documentation — sbert.net* [https://www.sbert.net/docs/sentence_transformer/pretrained_models.html]. [N.d.]. [Accessed 30-03-2026].

37. *Introducing gpt-oss* [https://openai.com/index/introducing-gpt-oss/]. 2025. [Accessed 20-04-2026].

38. *openai/gpt-oss-20b – Hugging Face* [https://huggingface.co/openai/gpt-oss-20b]. [N.d.]. [Accessed 04-05-2026].

39. HARPER, F. Maxwell; KONSTAN, Joseph A. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 2015, vol. 5, no. 4. ISSN 2160-6455. Available from DOI: 10.1145/2827872.

40. RESEARCH, GroupLens. *MovieLens 32M* [https://grouplens.org/datasets/movielens/32m/]. 2024. [Accessed 21-04-2026].

41. RESEARCH, GroupLens. *MovieLens* [https://grouplens.org/datasets/movielens/]. [N.d.]. [Accessed 21-04-2026].

42. *krishnakamath/movielens-32m-movies-enriched-with-SIDs – Datasets at Hugging Face — huggingface.co* [https://huggingface.co/datasets/krishnakamath/movielens-32m-movies-enriched-with-{S}{I}{D}s]. [N.d.]. [Accessed 21-04-2026].

43. WAN, Mengting; MCAULEY, Julian. Item recommendation on monotonic behavior chains. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 86–94. RecSys '18. ISBN 9781450359016. Available from DOI: 10.1145/3240323.3240369.

44. WAN, Mengting; MISRA, Rishabh; NAKASHOLE, Ndapa; MCAULEY, Julian. Fine-Grained Spoiler Detection from Large-Scale Review Corpora. In: KORHONEN, Anna; TRAUM, David; MÀRQUEZ, Lluís (eds.). *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 2605–2610. Available from DOI: 10.18653/v1/P19-1248.

45. *Goodreads Datasets — cseweb.ucsd.edu* [https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html]. [N.d.]. [Accessed 21-04-2026].

46. *About TMDB; The Movie Database (TMDB) — themoviedb.org* [https://www.themoviedb.org/about]. [N.d.]. [Accessed 21-04-2026].

47. ASANICZKA; THEMOVIEDB.ORG. *Full TMDB Movies Dataset 2024 (1M Movies)*. Kaggle, 2026. Available from DOI: 10.34740/KAGGLE/DSV/15852441.

48. HOU, Yupeng; LI, Jiacheng; HE, Zhankui; YAN, An; CHEN, Xiusi; MCAULEY, Julian. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*. 2024.

49. LAB, McAuley. *Amazon Reviews23* [`https://amazon-reviews-2023.github.io/`]. [N.d.]. [Accessed 21-04-2026].

50. JOULIN, Armand; GRAVE, Edouard; BOJANOWSKI, Piotr; MIKOLOV, Tomas. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759*. 2016.

51. JOULIN, Armand; GRAVE, Edouard; BOJANOWSKI, Piotr; DOUZE, Matthijs; JÉGOU, Hérve; MIKOLOV, Tomas. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*. 2016.

52. *Language identification · fastText — fasttext.cc* [`https://fasttext.cc/docs/en/language-identification.html`]. [N.d.]. [Accessed 22-04-2026].

53. *MSELoss; PyTorch 2.11 documentation — docs.pytorch.org* [`https://docs.pytorch.org/docs/2.11/generated/torch.nn.{M}{S}{E}{L}oss.html`]. [N.d.]. [Accessed 02-05-2026].

54. *python.org* [`https://www.python.org/doc/essays/blurb/`]. [N.d.]. [Accessed 20-04-2026].

55. *PyTorch documentation; PyTorch 2.11 documentation — docs.pytorch.org* [`https://docs.pytorch.org/docs/stable/index.html`]. [N.d.]. [Accessed 20-04-2026].

56. *pandas 3.0.2 documentation — pandas.pydata.org* [`https://pandas.pydata.org/docs/index.html#`]. [N.d.]. [Accessed 20-04-2026].

57. *Python module · fastText — fasttext.cc* [`https://fasttext.cc/docs/en/python-module.html`]. [N.d.]. [Accessed 20-04-2026].

58. *re – Regular expression operations — docs.python.org* [`https://docs.python.org/3/library/re.html`]. [N.d.]. [Accessed 20-04-2026].

59. *NumPy documentation; NumPy v2.5.dev0 Manual — numpy.org* [`https://numpy.org/devdocs/`]. [N.d.]. [Accessed 20-04-2026].

60. *train_test_split — scikit-learn.org* [`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`]. [N.d.]. [Accessed 20-04-2026].

Contents of the Attachment

```
/
├── README.md
├── autoencoders ..... Autoencoder training and testing source code
├── language_checking ..... English language testing
├── preprocessing ..... Jupyter notebooks for data preparation
├── summaries ..... Scripts for generating descriptions
├── text ..... Thesis
├── to_embeddings ..... Script for generating embeddings
├── utils ..... Helper functions
├── scenario_1 ..... Closed and Partially open scenarios code
└── scenario_2 ..... Open scenario code
```

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Beková** Jméno: **Linda** Osobní číslo: **500279**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra aplikované matematiky**

Studijní program: **Informatika**

Specializace: **Znalostní inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Identifikovatelnost položek z embeddingů založených na velkých jazykových modelech v doporučovacích systémech

Název diplomové práce anglicky:

Item Identifiability from Large-Language-Models-Based Embeddings in Recommender Systems

Jméno a pracoviště vedoucí(ho) diplomové práce:

Dr. rer. nat. Rodrigo Augusto Da Silva Alves katedra aplikované matematiky FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **04.02.2026** Termín odevzdání diplomové práce: **07.05.2026**

podpis vedoucí(ho) ústavu/katedry

podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Bc. Beková Linda

Podpis studentky

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Beková** Jméno: **Linda** Osobní číslo: **500279**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra aplikované matematiky**

Studijní program: **Informatika**

Specializace: **Znalostní inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Identifikovatelnost položek z embeddingů založených na velkých jazykových modelech v doporučovacích systémech

Název diplomové práce anglicky:

Item Identifiability from Large-Language-Models-Based Embeddings in Recommender Systems

Pokyny pro vypracování:

Recommender systems are a core component of modern online platforms, enabling personalized discovery across large item catalogs. In many machine learning-based recommender systems, side information is incorporated through dense embeddings, often generated by Large Language Models (LLMs), to represent items while abstracting away their original content. These embeddings are commonly treated as opaque representations that obscure the identity and descriptive information of items.
This project studies the identifiability of items from LLM-generated embeddings and investigates under which conditions item identity or content can be inferred. Several realistic scenarios will be considered, including cases where the set of items is known but item identifiers are not, where items are unknown, where the LLM used to generate embeddings is known or unknown, and where only partial text used to generate item embeddings is available. Analyzing these scenarios provides insights into the privacy, robustness, and interpretability of embedding-based recommender systems.
Proposed Tasks
1) Review existing literature on LLM-based item embeddings in recommender systems, with a focus on privacy, information leakage, and embedding inversion.
2) Formalize different item-identifiability scenarios based on the availability of item sets, identifiers, generating LLMs, and textual inputs.
3) Design and implement methods to infer item identity or content from embeddings under the defined scenarios (for at least two publicly available datasets).
4) Evaluate the proposed methods quantitatively and discuss the implications for privacy, security, and the practical deployment of embedding-based recommender systems.

Seznam doporučené literatury:

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY
Thákurova 9
160 00 Praha 6

DECLARATION

I, the undersigned

Student's surname, given name(s):	Beková Linda
Personal number:	500279
Programme name:	Informatics

hereby declare that the Master's Thesis titled

Item Identifiability from Large-Language-Models-Based Embeddings in Recommender Systems

is my own independent work and that I have declared all sources of information used in accordance with the Methodological Guideline for adhering to ethical principles when elaborating an academic final thesis and the Framework Rules for the use of artificial intelligence at CTU for study and pedagogical purposes in Bachelor and continuing Masters studies.

I declare that I have used artificial intelligence tools during the preparation and writing of the final thesis.
I've verified the generated content. I confirm that I am aware that I am fully responsible for the content of the final thesis.

In Prague on 07.05.2026

Bc. Linda Beková
.....
student's signature