杭州電子科技大学

本 科 毕 业 设 计（论 文）

（2023 届）

题　　目　个性化套装推荐系统开发

学　　院　

专　　业　

班　　级　

学　　号　

学生姓名　

指导教师　

完成日期　2023 年 6 月

诚 信 承 诺

我谨在此承诺：本人所写的毕业论文《个性化套装推荐系统开发》均系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释，若有不实，后果由本人承担。

承诺人（签名）：

2023 年 05 月 15 日

摘　要

本毕业设计旨在研究开发一种个性化时尚服装推荐系统，以帮助消费者更轻松地找到自己喜欢的服装。个性化套装推荐已经成为电商行业的一个热门研究方向，尽管时尚产业经济效益可观，但是互联网上的大量时尚产品使得在线购物者难以找到自己喜欢的服装，且全球时尚电商的规模还在持续高速增长，因此需要开发一个个性化服装偏好推荐系统，不仅能协助捕捉客户的时尚品味，还要能处理大规模数据，快速检索海量信息，提供准确和高质量的服装推荐服务。

现有的模型大多致力于研究如何提供更高质量的套装推荐而忽视了效率问题。其中时尚哈希网络模型兼顾了效率和效果，该模型使用神经网络分析视觉和文本信息，捕捉不同用户的喜好，并学习时尚产品之间的兼容性，并通过哈希技术实现高效的图像检索。尽管在大部分场景下时尚哈希网络能高效运行，但是在新用户冷启动场景，其效率仍有很大的改进空间。本文提出了一种针对新用户冷启动的基于相似度的用户编码算法，实验表明该算法在精度相差不大的情况下速度可以提升14倍，弥补了时尚哈希网络在冷启动场景下的短板。

最后基于该算法，本文开发了一个实时响应，在线推荐的网页系统，该网页系统能够为用户展示多个时尚套装的图片等信息，并且能根据用户的喜好快速调整更新推荐策略，提供符合用户喜好的个性化套装推荐。实验表明，该网页系统在个性化套装推荐问题上是快捷，有效的。

关键词：冷启动；神经网络；套装推荐；哈希学习；网页开发

ABSTRACT

The purpose of this thesis design is to research and develop a personalized fashion outfit recommendation system to help consumers find their favorite clothing more easily. Personalized outfit recommendation has become a popular research direction in the e-commerce industry. Despite the considerable economic benefits of the fashion industry, the large number of fashion products on the Internet makes it difficult for online shoppers to find their favorite clothes, and the size of global fashion e-commerce continues to grow at a high rate, so there is a need to develop a personalized outfit recommendation system that not only assists in capturing customers' fashion tastes, but also handles large scale data, quickly retrieve huge amount of information, and provide accurate and high quality clothing recommendation services.

Most of the existing models focus on how to provide higher quality outfit recommendations while neglecting the efficiency. Among them, the fashion hash network model balances efficiency and effectiveness. The model uses neural networks to analyze visual and textual information, capture the preferences of different users, and learn the compatibility between fashion products, and achieve efficient image retrieval through hashing techniques. Although the fashion hash network can work efficiently in most scenarios, there is still a large space for improvement in its efficiency in new user cold start scenarios. In this paper, we propose a similarity-based user encoding algorithm for new user cold start, and experiments show that the algorithm can improve the speed by 14 times with little difference in accuracy, which makes up for the shortcomings of the fashion hash network in cold start scenarios

Finally, based on this algorithm, this paper develops a time-responsive, online recommendation web system, which can display multiple images of fashion outfits and other information for users, and can quickly adjust and update the recommendation strategy according to the user's preferences, providing personalized outfit recommendations that match the user's preferences. The experiments show that the web system is efficient and effective in personalized outfit recommendation problem.

Key words： Cold Start; Neural Network; Outfit Recommendation; Learning to Hash; Web Development

目　　录

1 概述

1.1 引言

时尚推荐是一个巨大的产业。现在与 15 年前相比，平均每人购买 60%甚至更多的服装项目[1]。据报道，全球时尚的收入在 2018 年已经达到 4812 亿美元，与之相比，全球时尚电商收入有望迎来双倍增长，到 2023 年达到 8720 亿美元[2]。

尽管经济效益可观，但互联网上大量的时尚产品使网上购物者难以找到自己喜欢的服装。这促使我们开发一个准确和高质量的个性化服装偏好预测系统，协助捕捉客户的时尚品味，并进一步促进推荐服务。随着互联网购物平台（如亚马逊和淘宝）以及时尚相关的社交网络（如 Instagram）的快速发展，时尚推荐在日常生活中扮演着越来越重要的角色。它可以改善用户体验并为购物平台带来巨大的利润。据统计，个性化推荐已经影响了 43%的在线时尚购买，95%的在线时尚零售公司将个性化推荐视为一项重要而紧迫的商业战略[3]。

与传统的商品推荐相比，时尚套装推荐涉及到一个非常有创意的套装生成过程，它需要创新和特色。因此，这项任务通常由时尚专家来完成，并在众多在线时尚社区中流行，如 Lookbook[4]和 Chictopia[5]。中国最大的在线消费者对消费者平台淘宝网创建了一个新的应用程序 iFashion 以支持时尚的推荐。截至 2018 年 3 月 31 日，约有 150 万内容创作者积极支持淘宝网[6]。然而，有限的人力劳动和不断增长的市场之间的巨大差距，需要补充甚至替代人工工作。

综上所述，开发个性化套装推荐系统有着重大意义：

(1) 个性化套装推荐系统是电子商务行业中的一个热门研究方向，已经得到了广泛的关注。

(2) 随着用户对个性化服务的需求日益激增，推荐系统的研究和应用也越来越受到人们的重视。

(3) 个性化套装推荐系统的研究可以帮助电商平台提高用户购买率和满意度，从而增强企业的竞争力并提高市场份额

1.2 选题背景

以时尚为中心的社交网络已经成为一个充满活力的领域，数百万人在这里分享和发布日常时尚相关活动。这些社区的用户创造了大量的时尚服装。从这个庞大的数据集中挖掘出理想的时尚服装非常具有挑战性，但对这些在线时尚社区的发展至关重要。因此，人们非常需要智能的时尚推荐技术。每个服装类别中的商品数

[1] https://www.greenpeace.org/international/publication/6969/fashion-at-the-rossroads/
[2] https://www.mckinsey.com/industries/retail/our-insights/the-state-of-fashion
[3] https://www.shopify.com/enterprise/ecommerce-fashion-industry
[4] https://lookbook.nu/
[5] http://chictopia.com/
[6] https://www.alibabagroup.com/en/news/press_pdf/p180504.pdf

量的增长将导致可能的服装数量呈指数级增长。在实践中，服装的存储复杂性和检索效率对时尚推荐系统的部署至关重要，但这些问题在之前的工作中没能够被很好的处理。

近几年国内外的研究呈现出百家争鸣之势，已有的时尚推荐相关研究大致可分为以下几类：

1. 哈希 Learning to hash

Z. Lu 等人[1]设计了一个神经网络，用于高效的个性化服装推荐。它由三个部分组成。首先使用特征网络来提取内容特征。然后，一组依赖于类型的哈希模块将特征和用户口味表示转换为二进制代码。最后，通过进行成对加权哈希匹配的匹配块来计算用户对服装的总体偏好。他们的模型中使用了视觉和文本信息。

2. 自注意机制和纯思法模 Self-Attention and Transformer

Wen Chen 等人[2]通过从用户在时尚产品上的历史互动中捕捉用户的兴趣和品味来生成个性化的输出。对于兼容性要求，我们通过学习每个项目与输出中所有其他项目之间的兼容性，提出了一个时尚输出模型（Fashion Outfit Model，FOM）。每个项目应该与输出中的其他项目有不同的加权交互。基于自注意力机制建立了一个掩码项目预测任务，该任务一次屏蔽套中的一个项目，并根据套装中其他项目的上下文预测掩码项目。对于个性化需求，通过将用户偏好集成到预先训练的 FOM 中，我们提出了个性化输出生成（Personalized Outfit Generation，POG）模型，该模型可以基于用户的最近行为生成兼容的和个性化的输出。具体而言，POG 使用 Transformer 编码器-解码器架构根据用户偏好和外部兼容性对信号进行建模。

3. 图神经网络 Graph Neural Network

Xingchen Li 等人[3]明确地将用户、服装和物品之间的复杂关系呈现为一个层次化的时尚图。更具体地说，它由三个级别组成——用户、装备和物品级别——每个级别都包含相应类型的节点。与传统的图不同，这种分层图突出了跨级别的连接。此后，在分层时尚图上构建了一个新的框架，称为分层时尚图网络（Hierarchical Fashion Graph Network，HFGN）。特别地，HFGN 利用来自图神经网络（Graph Neural Networks，GNNs）的信息传播机制从下到上提取有用的信号，将关系注入表示中，并促进兼容性匹配和服装推荐。他们为每个用户/装备分配一个 ID 嵌入，同时用其视觉特征表示每个项目。信息传播规则聚合来自时尚物品的有用信号，以更新服装表示，并通过整合从他/她的历史服装传递的信息来进一步细化用户表示。此外还提出了一种联合学习方案，以同时进行兼容性匹配和服装推荐。

4. 多实例学习 Multiple Instance Learning

Yusan Lin 等人[4]将装备推荐任务转化为多实例学习（Multiple-Instance Learning，MIL）问题，开发了一个个性化的服装推荐系统 OutfitNet。OutfitNet 包括两个阶段：时尚物品相关性网络（Fashion Item Relevancy network，FIR）和套装偏好网络

（Outfit Preference network，OP）。FIR 学习给定时装项目子集、阳性项目和阴性样本的时装项目的相关性。OP 以不同数量的服装项目作为输入，并设计了一个个性化的关注层，以聚集时尚服装中的多个服装项目，以及学习个人在判断时尚服装时对服装项目的重视程度。

5. 知识图谱 Knowledge Graph

Zhan 等人[5]构建了一个新的注意力属性感知时尚知识图（Attentive Attribute-Aware Fashion Knowledge Graph，A^3-KFG），以捕捉实体节点之间的结构连接性，这些实体节点可以是不匹配的、项目和属性。此外，通过实体嵌入、从实体的连接邻居传播以及多模态时尚项目的内容特征（例如视觉图像和文本描述），服装表示得到了丰富。更具体地说，开发了一个关系感知注意力层来表征个人用户的细粒度兴趣，即服装的属性，这有助于学习服装表示。此外，用户的隐式和多样的口味是通过新的目标感知注意力层通过他/她先前交互的服装的加权聚合来建模的，这也是用户表示的一部分。最后，利用丰富的用户和服装表示来估计个性化偏好得分。知识图中由属性桥接的连接和注意力网络结构共同有助于解释预测。

1.3 系统概述

时装库存中的商品数量通常非常多，而这些商品所能组成的服装数量要大几个数量级。想要部署一个实用的时尚推荐系统，效率是一个不容忽视的问题。物品和配件应以经济的方式储存。一套服装的兼容性不仅要准确评估，而且要有效评估。而且最兼容的物品和服装需要迅速找回。这些问题在以前的作品中没有得到很好的处理。在这项工作中，我们将它们考虑在内，并探索时尚推荐的哈希技术。学习散列对于高效的图像检索已经进行了广泛的研究[6]。

我们设计了一个神经网络，用于高效的个性化时尚服装推荐。该网络捕捉不同用户的喜好，并学习时尚物品之间的兼容性。它由三个部分组成。首先，使用特征网络来提取物品的内容特征。然后，一组类型相关的哈希模块将特征和用户喜好表示转换为二进制代码。最后，通过进行成对加权哈希匹配的匹配块来计算用户对服装的总体偏好。我们的模型同时使用了视觉和文本信息。由于之前工作使用的数据集要么数据量太少，要么缺乏用户信息，我们收集了一个新的数据集，其中包含数百名在线用户创建的 138,000 多套服装。我们利用这个大规模数据集，对我们的方法进行了评估。大量实验表明，即使使用简单的主干和二进制表示，它也优于最先进的方法。

当聚焦于新用户冷启动这一特定场景下，我发现神经网络算法存在一些不足。因为该神经网络对用户采用 one-hot 编码策略，因此无法直接对新用户进行哈希编码计算，其中一种方法是重新训练神经网络，虽然此举在性能表现上是优秀的，但是在效率是难以接受的：（1）需要强悍的硬件设备，即图形处理器（Graphics Processing Unit，GPU）支持；（2）需要一定量的数据支持训练；（3）训练一个神

经网络所耗费的时间是很久的。因为受限于以上的种种限制，所以我另辟蹊径，提出了一种基于相似度的用户编码算法，其包含两个部分（1）基于用户相似度的编码：根据新用户与老用户的相似度以及老用户的用户编码来计算新用户的用户编码；（2）基于物品相似度的编码：因为用户编码和物品编码存在相关性，根据用户喜欢或不喜欢的套装物品编码来计算用户的编码。实验表明，基于相似度的用户编码算法在冷启动问题上的速度远远快于基于神经网络的算法，并且准确度并没有落后太多。

最后基于相似度的用户编码算法，本文开发了一个时响应，在线推荐的网页系统。网页系统能够为用户展示多个时尚套装，每个套装包括所有类别（上衣，裤子，鞋子）的图片信息，网页将三张图片自上而下并列展示。并且在子页面将详细信息一一展示，提供两个按钮，分别是“喜好”和“不喜欢”，允许用户表达自己的喜好，点击按钮后系统会记录用户的操作信息并在后端进行更新。刷新页面，系统将会根据用户的喜好快速调整更新推荐策略，重新计算，为用户推荐新的一组套装。实验表明，该网页系统在个性化套装推荐问题上是快捷，有效的。

图 1-1　三个用户的推荐服装示例，红色框中为一个时尚套装中所有服装图片（图片引自[1]）

1.4 论文组织结构

本论文的组织结构可以大致分为以下六个部分：概述部分、算法设计部分、网页开发部分、实验部分以及总结与展望部分。

(1) 概述部分

论文的概述部分介绍了本课题的研究现状，即本课题研究的内容与意义，以及论文的组织结构。概述部分扼要介绍了系统涉及到的各个方面。

(2) 算法设计部分

论文的算法设计部分介绍了本设计涉及的个性化套装推荐系统总体构造，以及针对冷启动的改进和评估方法。

(3) 网页开发部分

论文的网页开发部分介绍了个性化套装推荐系统中网页开发部分的内容，包括后端如何对根少量信息对新用户进行个性化套装推荐，如何处理用户的交互信息；以及前端如何编写网页代码来展示个性化套装，如何提供用户交互接口。

(4) 实验部分

论文的实验部分介绍了实验的环境、数据集信息以及算法和前后端在编写代码和完成后进行的测试评估结果。

(5) 总结与展望部分

论文总结与展望部分介绍了本设计的完成情况以及对未来可扩展功能的展望。以及本人对完成毕业设计的一些感想与感悟

2 算法设计

2.1 问题定义

与传统的新闻或电影推荐类似，以前的研究大多集中在向给定的用户推荐单一的时尚物品。然而，在实践中，用户真正需要的通常是一套匹配度高的服装（即一组兼容和互补的时尚物品），而不是单一的时尚物品。

假设有N时尚类别（例如上衣，裤子，鞋子）。第n个类别中的物品数量表示为L_n，用户数量为U。设$X^{(n)} = \{x_1^{(n)}, \dots, x_{L_n}^{(n)}\}$表示第$n$个分类中的所有物品，其中$x_i^{(n)}$是其中的第$i$个项目。然后，一个套装有$N$个项目，每个项目来自一个类别，表示为：

$$\mathcal{O}_i = \{x_{i_1}^{(1)}, x_{i_2}^{(2)}, \dots, x_{i_N}^{(N)}\}. \quad (1)$$

其中$i = (i_1, \dots, i_N)$是索引元组。

不同用户的时尚品味可能非常不同，即时尚风格偏好是个人的。特定用户的个性化时尚偏好可以通过用户过去创建或评分的服装来隐含地表示。我们使用$r_{u,\mathcal{O}_i}$来表示用户u对$\mathcal{O}_i$的偏好。得分越高，用户越喜欢这套衣服。我们的任务是预测任何用户和服装对的$r_{u,\mathcal{O}_i}$，以便向用户推荐最优选的服装。

在不失一般性的情况下，让我们假设一套服装由N个类别的N个物品组成。那么可能的用户套装总数是$U \times L1 \times \cdots \times LN$，这在实践中是一个非常大的数字。因此，我们需要以更可行的方式计算$r_{u,\mathcal{O}_i}$，以便应用一些有效的搜索技术。在这项工作中，我们探索了二进制嵌入技术，即我们试图用二进制代码来表示用户和时尚项目，从中计算偏好分数。

2.2 总体设计

Zhi Lu 等人[1]提出了一种时尚哈希网络（Fashion Hashing Network，FHN）来预测用户对不同服装成分的偏好。总体架构如图 3-1 所示。它由三种类型的组件组成，一个用于特征提取的特征网络，几个用于学习二进制码的类型相关哈希模块，以及一个用于预测偏好得分的匹配块。对于输入，每个用户由 one-hot 向量表示，表示用户的索引。我们使用卷积网络来提取图像的视觉特征。不同项目类别的要素网络共享相同的参数。这个特征网络的具体结构对我们来说是可选的。请注意，文本信息也可以合并，我们将在稍后讨论。来自不同类别和用户的项目被视为不同的类型。

每个散列模块由一些完全连接的层组成，这些层具有用于二进制化的符号函数。最后一个组成部分是匹配块，用于计算给定二进制码的偏好得分。有两个术语对最终得分有贡献。一个只考虑物品的兼容性，另一个则考虑用户的品味。详细的配方将在以下小节中进行讨论。

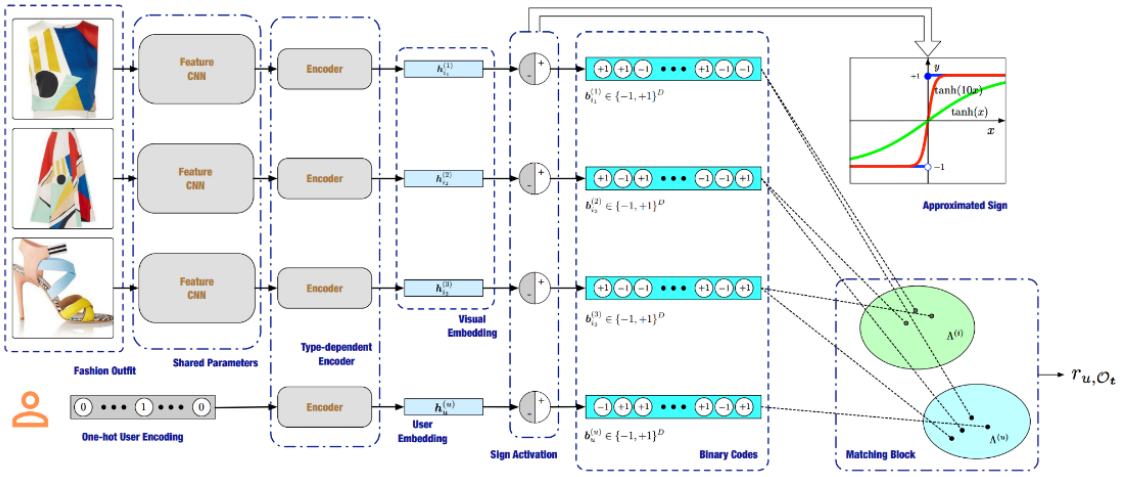


图 3-1　总体架构（图片引自[1]）

2.2.1　匹配块

服装物品之间的兼容性和对用户的适合度是多向关系。尽管在各种应用中已经对高阶关系船舶模型做出了许多努力，但最成功的处理方法仍然是将其分解为成对关系船舶，这更容易学习，并且已被证明是一种很好的近似方法。考虑到计算偏好得分和检索兼容服装进行推荐的效率，我们还建立了基于成对交互关系的模型。

令$b_i, b_j \in \{-1,+1\}^D$ 是两个对象的二进制编码，这两个对象可以是时尚项目或用户，它们的兼容性通过以下公式测量

$$m_{ij} = \boldsymbol{b}_i^{\top} \boldsymbol{\Lambda} \boldsymbol{b}_j, \tag{2}$$

其中Λ是一个加权矩阵，我们将其约束为对角线，即$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_D)$。引入$\Lambda$是为了更好地捕捉对象之间的关系，同时主要保持二进制编码的计算效率。并且服装$\mathcal{O}_i$相对于用户u的得分通过以下公式计算

$$r_{u,O_i} = \alpha \cdot r_{u,O_i}^{(u)} + r_{u,O_i}^{(i)}, \tag{3}$$

其中

$$r_{u,O_i}^{(u)} = \frac{1}{z_1} \sum_n \boldsymbol{b}_{i_n}^{(n)\top} \boldsymbol{\Lambda}^{(u)} \boldsymbol{b}_u^{(u)}, \tag{4}$$

$$r_{u,O_i}^{(i)} = \frac{1}{z_2} \sum_n \sum_m \boldsymbol{b}_{i_n}^{(n)\top} \boldsymbol{\Lambda}^{(i)} \boldsymbol{b}_{i_m}^{(m)}. \tag{5}$$

在上述公式中，$\boldsymbol{\Lambda}^{(u)}$和$\boldsymbol{\Lambda}^{(i)}$分别是用户-项目和项目-项目对的权重矩阵。式（4）模拟了用户对时尚单品的偏好，式（5）测量了成对的时尚物品之间的兼容性。它们通过所涉及的套装对数（即z_1和z_2）进行归一化。标量α用于平衡这两个项的贡献。尽管α可以被吸收到$\boldsymbol{\Lambda}^{(u)}$中，但我们将其保留在式（3）用于稍后不使用加权哈希的讨论，即$\boldsymbol{\Lambda}^{(u)}, \boldsymbol{\Lambda}^{(i)}$被设置为单位矩阵$\boldsymbol{I}$。

2.2.2　哈希学习

直接优化具有$\boldsymbol{b}_{i_n}^{(n)}, \boldsymbol{b}_u^{(u)}$二元约束的目标函数是具有挑战性的。在优化过程中，

通过使用连续变量来替换它们来重新松弛问题是很常见的。二进制码是通过取连续变量的符号获得，例如

$$\boldsymbol{b}_{i_n}^{(n)} = \text{sign}(\boldsymbol{h}_{i_n}^{(n)}), \boldsymbol{b}_u^{(u)} = \text{sign}(\boldsymbol{h}_u^{(u)}), \tag{6}$$

其中$\text{sign}(x) = 1 \text{ if } x \geq 0 \text{ and } -1 \text{ otherwise}$。

许多散列方法通过最小化二进制化后的量化误差来学习散列码。最近，Cao 等人[7]提出了一种 HashNet 方法，该方法通过在神经网络中激活来近似 sign 运算。这种方法避免了明确地处理量化误差。在这项工作之后，我们将式（4）和式（5）重写如下：

$$r_{u,O_i}^{(u)} = \frac{1}{z_1} \sum_n \hat{\boldsymbol{b}}_{i_n,t}^{(n)\top} \boldsymbol{\Lambda}^{(u)} \hat{\boldsymbol{b}}_{u,t}^{(u)}, \tag{7}$$

$$r_{u,O_i}^{(i)} = \frac{1}{z_2} \sum_n \sum_m \hat{\boldsymbol{b}}_{i_n,t}^{(n)\top} \boldsymbol{\Lambda}^{(i)} \hat{\boldsymbol{b}}_{i_m,t}^{(m)}, \tag{8}$$

其中

$$\hat{\boldsymbol{b}}_{i_n,t}^{(n)} = \tanh\left(\beta_t \boldsymbol{h}_{i_n}^{(n)}\right), \hat{\boldsymbol{b}}_{u,t}^{(u)} = \tanh\left(\beta_t \boldsymbol{h}_u^{(u)}\right), \tag{9}$$

并且β_t是在优化过程中随着迭代t而增加的标量。当β_t较大时，式（9）是式（6）的近似值（见图 3-1 中的图表）。因此，当优化结束时，$\hat{\boldsymbol{b}}_{i_n,t}^{(n)}$和$\hat{\boldsymbol{b}}_{u,t}^{(u)}$，t 收敛到二进制编码$\boldsymbol{b}_{i_n,t}^{(n)}$和$\boldsymbol{b}_{u,t}^{(u)}$。

2.2.3 目标函数

除了图片之外，时尚单品在网上展出时通常也会被一些文字描述所描绘。这些文本描述为图像提供了语义信息，这对兼容性建模非常有帮助。因此，我们还使用了从输入输出模型中的文本内容中提取的特征。

我们使用相同的方法将文本特征转换为二进制代码，并计算相应的偏好分数。假设来自不同模态的二进制代码由$\boldsymbol{b}_{v,i_n}^{(n)}$和$\boldsymbol{b}_{f,i_n}^{(n)}$提供，其中$v$和$f$分别表示 visual 和 text。通过将每个模态的得分相加来计算具有多模态信息的总得分：

$$r_{u,O_i}(\{\boldsymbol{b}_{v,i_n}^{(n)}\}, \boldsymbol{b}_u^{(u)}) + r_{u,O_i}(\{\boldsymbol{b}_{f,i_n}^{(n)}\}, \boldsymbol{b}_u^{(u)}). \tag{10}$$

每个模态的得分由式（3）计算，并且用户的相同二进制编码用于不同模态。

我们使用 ranking loss 来学习我们模型的参数，即训练网络，目的是当给定一套服装的情况下，它可以预测哪一套比用户更可取。训练套装包含一系列套装对：

$$\mathcal{P} \equiv \{(u, \boldsymbol{i}, \boldsymbol{j}) | r_{u,O_i} > r_{u,O_j}\}, \tag{11}$$

其中$(u, \boldsymbol{i}, \boldsymbol{j})$指示用户$u$更喜欢服装$O_i$而不是$O_j$。我们采用 BPR[8]优化准则来最大化模型参数的后验概率。目标可以表示为：

$$\ell_{BPR} = \sum_{(u,\boldsymbol{i},\boldsymbol{j}) \in \mathcal{P}} \log\left(1 + \exp(-(r_{u,O_i} - r_{u,O_j}))\right), \tag{12}$$

继之前的工作[9]之后，我们还添加了约束，以使视觉和文本信息的嵌入更加一致。为了简单起见，假设$\boldsymbol{v}, \boldsymbol{f}$是来自两种模态的项目的二进制编码。它们的相似性

是由$s(\boldsymbol{v},\boldsymbol{f})=\boldsymbol{v}^{\top}\boldsymbol{f}$提供的。我们要求对应于同一项目的$\boldsymbol{v},\boldsymbol{f}$应该比对应于不同项目的更相似。这是通过将以下损失降至最低来实现的：

$$\ell_{VSE}=\sum_{\boldsymbol{v},k}\max\left\{0,c-s(\boldsymbol{v},\boldsymbol{f})+s\left(\boldsymbol{v},\boldsymbol{f}_k\right)\right\}+\sum_{\boldsymbol{f},k}\max\left\{0,c-s(\boldsymbol{v},\boldsymbol{f})+s\left(\boldsymbol{v}_k,\boldsymbol{f}\right)\right\},\tag{13}$$

其中$(\boldsymbol{v},\boldsymbol{f})$用于相同的项目。$(\boldsymbol{v},\boldsymbol{f}_k)$用于不同的项目，$(\boldsymbol{v}_k,\boldsymbol{f})$也是如此。

总体而言，所提出方法的目标是：

$$\min_{\Theta}\mathbb{E}(\ell_{BPR}+\lambda\ell_{VSE}).\tag{14}$$

其中，Θ是网络的参数，λ是加权参数。Θ包括$\Theta_{\mathrm{nn}},\Theta_v,\Theta_f,\Theta_u$，其中$\Theta_{\mathrm{nn}}$是特征网络的参数，而$\Theta_v,\Theta_f,\Theta_u$分别是两种模态和用户的编码器的参数。如第 3.2.2 节所述，通过连续松弛来解决优化问题。

2.3 时尚哈希网络的冷启动算法

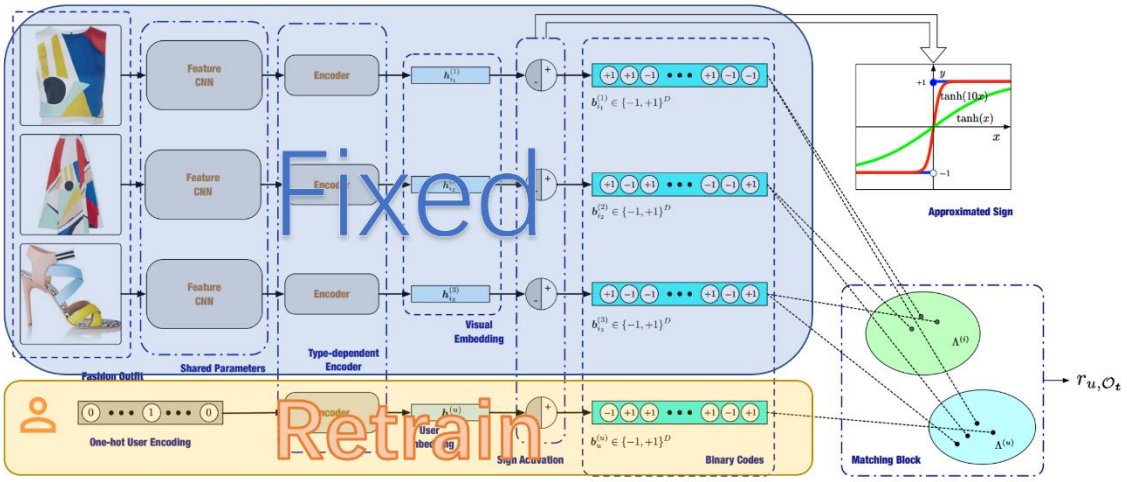


图 3-2 基于神经网络的冷启动算法

推荐系统中的冷启动问题是指系统在缺乏用户数据或物品数据的情况下无法进行有效的推荐。这可能是由于新上线的产品、新注册的用户或者系统数据不足等原因导致的。冷启动问题会影响到推荐系统的准确性和实用性，因为推荐系统需要大量的数据才能建立个性化的推荐模型。

新用户不断加入社交网络，为每个新用户重新培训整个网络将是不可承受的。为了解决这个问题，时尚哈希网络（FHN）保持项目的特征网络是固定的，并且只重新训练新用户的用户表示，这在方法中只是一个 128 位的二进制代码，可以非常有效地计算。他们评估了一个场景，其中系统已经为过去的 630/529 用户建立了一个模型，以及 53/32 个新用户。与 Polyvoe-630/519 上的性能相比，性能下降是可以接受的，即使数据集的大小增长了 7%左右，该方法也只能通过微调新用户的表示来保持性能。

2.4 冷启动算法改进

Lu Z 等人[1]基于重新训练神经网络的冷启动算法虽然性能优异，但是其缺陷也很明显：（1）需要强悍的硬件设备（即 GPU）支持；（2）需要一定量的数据支持训练；（3）训练一个神经网络所耗费的大量时间是难以接受的。因此该算法难以达到在线运行，实时推荐的要求。

为了解决上面所描述的效率问题，我所选择的解决方法是换用一种基于相似度的用户哈希编码生成算法，该算法的速度远远快于基于重新训练的算法，适用于后续的网页开发。

2.4.1 基于相似度编码

我打算放弃神经网络，而采用传统的方法求解新用户的二进制编码。而用户编码的计算分为两部分。（1）利用老用户的编码来计算新用户的编码。（2）计算新用户对物品的喜好来计算新用户的编码。

$$\boldsymbol{b}_u = \beta \cdot \boldsymbol{b}_{u-user} + (1-\beta) \cdot \boldsymbol{b}_{u-item}. \tag{15}$$

其中标量β用于平衡这两个项的贡献。

2.4.2 基于用户相似度编码

首先需要计算新用户和老用户的相似度，如果相似度越大，那么两者的编码应该也会越近似，反之则反。受式（5）的启发，我通过两个用户喜欢或不喜欢的物品编码来计算两个用户的相似度。定义两个套装序列的相似度为

$$s_{O_u^{posi},O_{u'}^{posi}} = \frac{1}{L_u^{posi} L_{u'}^{posi}} \sum_i^{L_u^{posi}} \sum_j^{L_{u'}^{posi}} \frac{1}{ND} \sum_n^N \boldsymbol{b}_{u_i^{posi}}^{(n)\top} \boldsymbol{\Lambda}^{(i)} \boldsymbol{b}_{u'^{posi}_j}^{(n)}, \tag{16}$$

其中O_u^{posi}表示用户u喜欢的所有套装列表，反之O_u^{nega}表示用户u不喜欢的所有套装列表。那么两个用户之间的相似度就可以表示为

$$r_{u,u'} = \frac{r_{O_u^{posi},O_{u'}^{posi}} - \epsilon(r_{O_u^{nega},O_{u'}^{posi}} + r_{O_{u'}^{nega},O_u^{posi}})}{1+2\epsilon}, \tag{17}$$

其中标量ϵ用于平衡正负样本的贡献，因为数据集包含了正样本，而负样本则通过随机或者取他人的正样本生成，因为我认为应该降低负样本的权重，即$0 < \epsilon < 1$。

最后，通过老用户的编码计算新用户的编码

$$\boldsymbol{b}_{u-user} = \frac{1}{U} \sum_i^U r_{u,u_i} \boldsymbol{b}_u^{(i)}. \tag{18}$$

2.4.3 基于物品相似度编码

受式（5）的启发，可以通过新用户喜欢或不喜欢的物品的编码来计算用户编码，其目标为找到合适的用户编码使得对这些物品的偏好分数尽可能大，即

$$\begin{aligned} &\underbrace{\arg\max}_{\boldsymbol{b}_{u-item}} r_{u,O^{posi}} = \frac{1}{L^{posi}} \sum_i^{L^{posi}} r_{u,O_i^{posi}}, \text{ and} \\ &\underbrace{\arg\min}_{\boldsymbol{b}_{u-item}} r_{u,O^{nega}} = \frac{1}{L^{nega}} \sum_i^{L^{nega}} r_{u,O_i^{nega}}, \end{aligned} \tag{19}$$

等价于

$$\underbrace{\arg\max}_{\boldsymbol{b}_{u-item}} \frac{r_{u,O^{posi}}-r_{u,O^{nega}}}{2}, \tag{20}$$

解得

$$\boldsymbol{b}_{u-item} = \frac{1}{2DN} \cdot \mathrm{sign}(\boldsymbol{\Lambda}^{(u)}) \left(\frac{1}{L^{posi}} \sum_{i}^{L^{posi}} \sum_{n}^{N} \boldsymbol{b}_{i_n^{posi}}^{(n)} - \frac{1}{L^{nega}} \sum_{i}^{L^{nega}} \sum_{n}^{N} \boldsymbol{b}_{i_n^{nega}}^{(n)} \right). \tag{21}$$

详细的推导过程见附录。

2.4.4 时间复杂度分析及优化

基于用户相似度编码的时间复杂度为$O(UL^2ND)$，基于物品相似度编码的时间复杂度为$O(LND)$，因此基于相似度的编码总体时间复杂度为$O(UL^2ND)$。

通过观察不难发现，基于用户和基于编码的两部分计算内容中有重叠区域，即物品编码运算部分，可以将该部分提取出来并离线计算出来，即提前计算出每个用户喜欢或不喜欢的套装序列所有物品的编码均值，详细的推导过程见附录。

通过这种操作，离线计算的时间复杂度为$O(LND)$，在线计算部分中，基于用户相似度编码的时间复杂度为$O(UD)$，基于物品相似度编码的时间复杂度为$O(D)$，总的在线计算时间复杂度为$O(UD)$。这样的时间复杂度对于在线的网页系统来说是可以接受的，并且不需要 GPU 支持，该算法可以只依赖计算机中央处理器（Central Processing Unit，CPU）就可以高效运行。

3 网页开发

3.1 后端

本文使用 Python 编写了网页的后端，并采用了 Flask[7] 框架。Flask 是一个基于 Python 编写的轻量级 Web 应用程序框架，它提供了许多工具和功能，使得 Web 应用程序的开发变得非常简单和灵活。Flask 具有灵活的架构和丰富的扩展库，可以快速地构建高效和安全的 Web 应用程序。因此，使用 Flask 作为本文的网页开发框架是非常合适的。

3.1.1 文件存取

为了保证网页系统的流畅性，且受限于硬件条件的限制，本文并打算在后端使用神经网络相关的算法，而是在网页启动之前，使用时尚哈希网络（FHN）计算出所有套装物品的编码以及其他必要参数信息并保存到本地文件，包括超一百万套个性化套装对应的物品编码和物品图片文件地址。我将编码压缩后存储，文件约 130MB。

在启动后端的初始化过程中，从本地文件中加载数据，并将数据存放到内存中。为了高效快捷得完成数据存取的操作，我利用了 pickle[8] 模块。pickle 模块是 Python 标准库中用于序列化和反序列化 Python 对象的模块之一。它可以将 Python 对象转换为二进制格式以便于在网络传输或保存到本地磁盘中，也可以反序列化二进制数据以重建原始的 Python 对象。pickle 模块支持几乎所有的 Python 对象类型，包括自定义的类和函数等。使用 pickle 模块可以方便地进行对象的持久化和传输，使得 Python 程序更加灵活和高效。

3.1.2 套装推荐

当页面刷新时，后端会计算要推荐的套装序列，将套装的图片地址和套装编号传递到前端。

如何计算推荐的套装序列？我枚举所有套装，依据式（15）计算出该用户对每件套装的偏好分数，并按照从大到小依次排序。其中的物品编码信息在初始化阶段已经从文件加载到内存中，用户编码在初始化为随机编码，并在之后根据用户交互信息进行更新。

假设前端一次能显示的套装数量为K，为了兼顾推荐的多样性，我并没有直接取排名最高的K个套装，而是在前$10K$个套装中随机采样K个传递给前端。这样也避免了用户没有找到心仪的套装多次刷新却给出相同推荐的问题。这里我们取K=10，具体算法流程见 Algorithm 1.

[7] https://flask.palletsprojects.com/
[8] https://docs.python.org/3/library/pickle.html

Algorithm 1 Recommend Outfits for User

Input: user_code, Outfits;
Output: recommend_outfits

```
ratings=[]
for outfit in Outfits do
    ratings.append(calculate_rating(outfit, user_code)) // ref Eq.(3)
end for
sort(Outfits, key=ratings, reverse=True)
recommend_outfits = random.sample(Outfits[:100], 10)
return recommend_outfits
```

3.1.3 交互响应

当用户在前端做出点击按钮操作，其对某件套装的喜欢或不喜欢的数据会传递到后端，而后端则需要记录该信息并利用这条信息对用户编码进行更新。

我定义了两个序列分别用来存储用户喜欢的套装信息和用户不喜欢的套装信息，根据用户的点击信息分别将套装信息存储到对应的序列中去。

更新用户编码采用的算法如式（13）所示。整个点击交互响应的具体流程算法见 Algorithm 2。

Algorithm 2 Update User Encoding on Click

Input: action, outfit_id, clicks_posi, clicks_nega;
Output: user_code

```
if action == 'like' then
    clicks_posi.append(outfit_id)
end if
if action == 'dislike' then
    clicks_nega.append(outfit_id)
end if
user_code=calculata_user_embedding(clicks_posi, clicks_nega) //ref Eq.(15)
return user_code
```

3.2 前端

我从 HTML5 UP[9]网站下载了 Multiverse 网页模板，并在此基础上做出修改以符合个性化套装推荐系统网页开发的需求。

3.2.1 套装显示

每个套装通常由三个类别组成，上衣，裤子，鞋子。每个类别分别对应一张图片，我们需要将一个套装里的所有物品统一展示，并且最好要满足不同类别匹配身上不同的部位，即三张图片要自上而下显示。由于以上所述套装的特性，我决定在主页面，网页每行显示五个套装，显示两行一共十个套装，如图 4-1 所示。且每个套装下面加上标题，显示内容为服装的编号。

技术上为了实现这个效果，我修改了 css 文件使得每个套装的模块（article）宽度为 20%。为了使套装中的三张图片有序的显示在一起，创建了对应的 class。设置其参数，其中长度宽度为图片尺寸，设置 float: left，并调整到居中。

[9] https://html5up.net/

另外，考虑到网页中出现了大量的图片元素，尽管后端的效率足够高，前端的图片加载仍然存在不小的负担。为此我使用了图片懒加载技术：图片懒加载是一种优化网站性能的技术，它通过推迟网页中图片的加载，直到它们出现在用户的视窗内才开始加载，从而减少了初始页面加载时间和带宽使用。懒加载可以节省页面加载时间，减少资源消耗，提高用户体验。特别是对于长页面或包含大量图片的页面，懒加载可以有效地提高页面加载速度和性能。

图 3-1 网页主界面，展示十个套装。每个套装包含上衣，裤子，鞋子三张图片。

3.2.2 按钮交互

我们在主页面只展示套装的图片和标题，以提供沉浸式的视觉体验。如果用户对某个套装感兴趣，那么通过点击该套装图片的任意处即可进入子页面。子页面内展示该套装的图片信息和详细描述，并提供“喜欢”和“不喜欢”两个按钮。如图 3-2 所示。

用户可以通过点击按钮来表示自己的喜好。当鼠标移到“喜欢”按钮上时，按钮会变为绿色。点击后，信息会发送到后端进行处理。当处理完成后，会弹出提示框显示“你喜欢这个套装”，如图 3-3 所示。同样的，当鼠标移到“不喜欢”按钮上时，按钮会变为红色。点击后经过相同的步骤并在最终跳出提示框显示“你不喜欢这个套装”。这样的交互逻辑能够给用户清晰的路径和反馈。

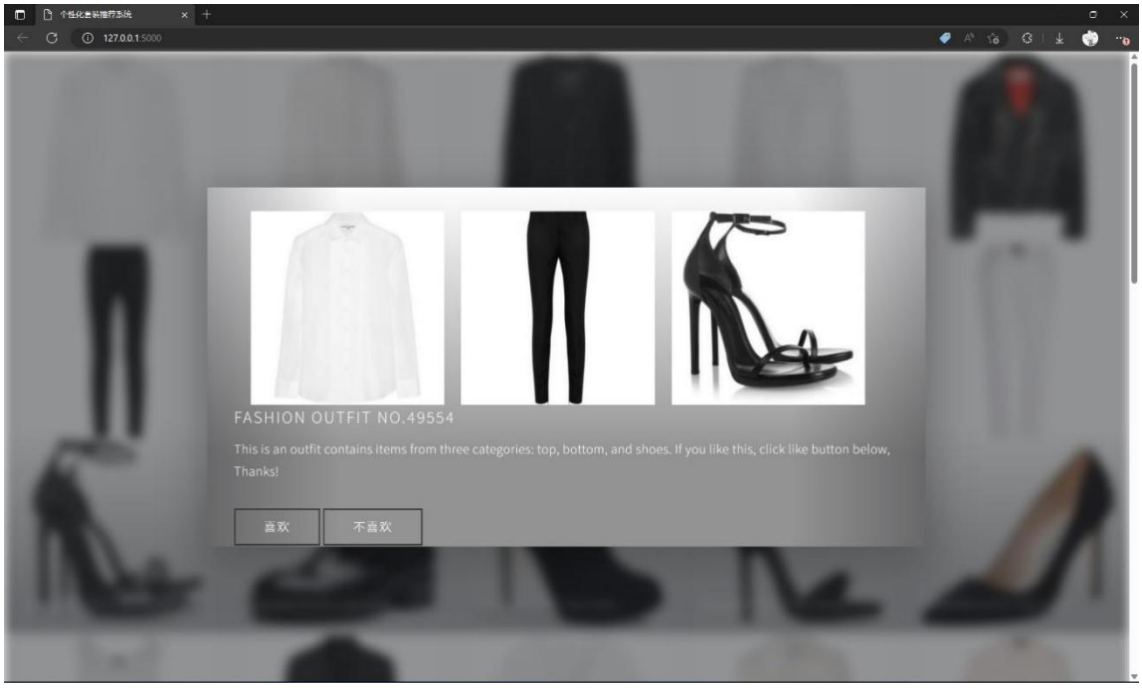


图 3-2 网页子界面

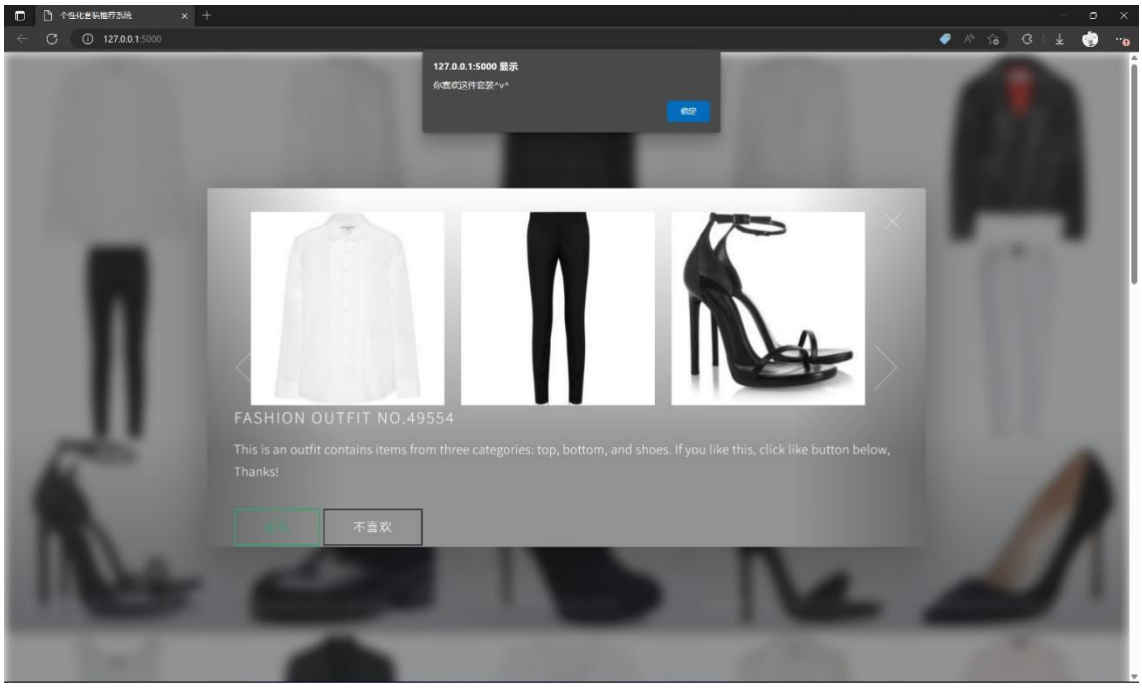


图 3-3 点击“喜欢”按钮

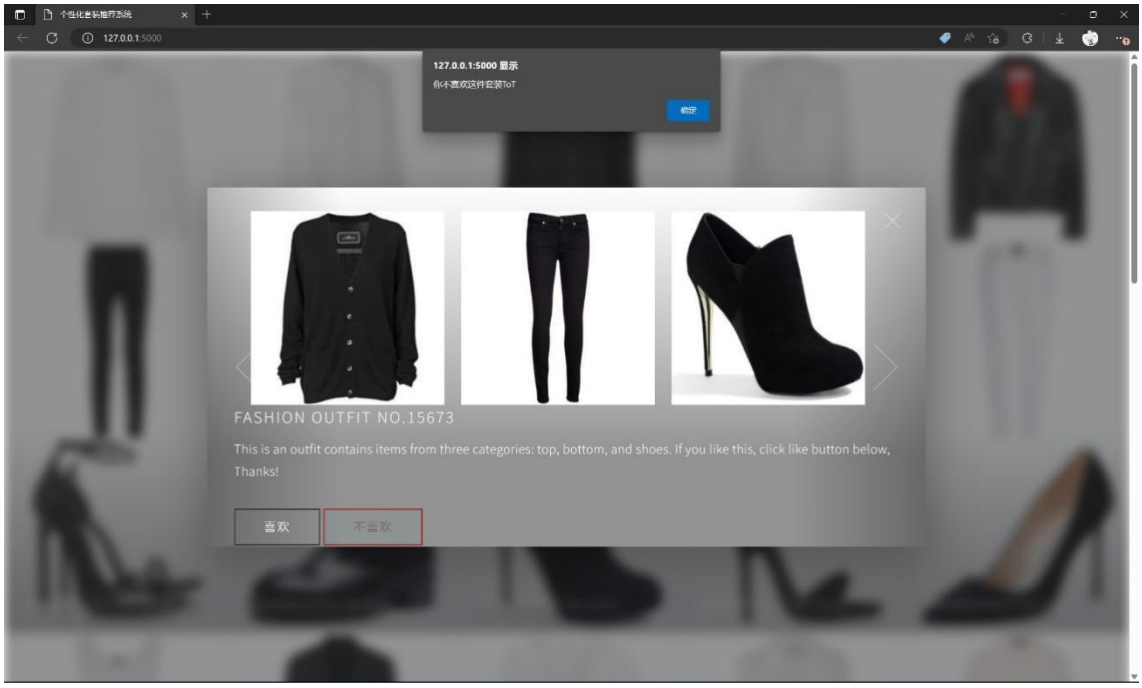


图 3-4 点击“不喜欢”按钮

技术上为了实现这个效果，我们定义了一个脚本函数来执行点击按钮后的操作。该函数接受动作（喜欢或不喜欢）和套装编号，将这些信息通过 POST 发送到后端，最终调用 alert 函数来输出提示表明后端已经接受并完成了相关操作。

3.2.2 网页刷新

按钮交互并不会刷新页面或者重新做推荐，防止突然刷新使用户错过喜欢或不喜欢的套装，这样和符合主流推荐系统的逻辑。因此我的网页通过用户主动刷新来进行套装推荐的更新，每次刷新后后端会根据最新的用户编码重新推荐新的十个套装。

如果用户没有点击任何按钮，即用户编码并没有更新，网页也不会提供你相同的套装推荐，而是会很大概率刷新页面后推荐了不一样的十个套装。因为我在后端算法中对所有套装进行了一定程度的随机采样，具体细节见第 3.1.2 节。这样的设计能够为用户提供更多的选择。

4 实验结果

4.1 实验环境

在本文中使用了两种实验环境：

(1) 算法设计硬件环境：

我使用了一台操作系统为 Ubuntu 16.04.1，CPU 型号为 Intel i7-9800X，GPU 型号为 NVIDIA TITAN RTX 的服务器。

(2) 网页开发硬件环境：

我使用了一台操作系统为 Windows 11，CPU 型号为 AMD Ryzen 5 5600H 的笔记本。

本文所有工作的相关代码已经全部上传至 GitHub[10]。

4.2 数据集

4.2.1 Polyvore-U

我使用了 Zhi Lu[1] 提出的数据集 Polyvore-U。由于现有的数据集[9–11]要么太小，要么缺乏用户信息，因此这些数据集难以被用来为我们所研究个性化时尚服装推荐问题评估。于是他们从 Polyvore 网站收集了一个新的数据集。让每件衣服包含三类物品，即上衣、下装和鞋子，创建了 4 个版本的数据集，用 Polyvore-U 表示，其中 U 是用户数量。两个数据集，例如 Polvvore-630 和 Polyvore-53，包含具有固定数量物品的服装，即每个服装在每个类别中只有一个物品。在其他两个数据集中，Polvvore-519 和 Polyvore-32，每件衣服中的物品数量是不同的，即有些衣服可能有两件上衣。Polvvore-630 和 Polvvore-519 这两个较大的数据集被用于大多数实验。Polyvore-53 和 Polyvore-32 是为了测试模型的用户泛化能力而保留的。数据集的统计数据如表 4-1 所示。

表 4-1 Polyvore 数据集的统计信息

数据集	切分	训练集	测试集
Polyvore-630	#Outfits	127,326	23,054
	#Items	159,729	45,505
Polyvore-53	#Outfits	10,712	1,944
	#Items	20,230	4,437
Polyvore-519	#Outfits	83,416	14,654
	#Items	146,475	39,085
Polyvore-32	#Outfits	5,133	898
	#Items	14,594	2,797

[10] https://github.com/KaizynX/Fashion-Hash-Net

4.2.2 数据准备

我们将用户创建的服装视为该用户的相关服装，即正样本。他/她的不相关服装，即负样本来自两个方面。一种是物品的随机混合，另一种是其他用户正面服装的随机样本。第二种类型的负面服装比第一种更难。不考虑个性化问题的服装组合方法通常无法将其与正面服装区分开来。每个用户的不相关服装和相关服装之间的比例设置为 10:1。在每个数据集中，消极服装中的项目数量与积极服装中的数量保持一致。我们确保每个用户的培训和测试集之间没有项目重叠。

对于冷启动算法，我使用了 Polyvore-630 数据集训练模型，并使用 Polyvore-53 数据集验证模型对新用户的个性化套装推荐效果。同时还测试了在少量数据的情况下基于神经网络的算法和基于相似度的算法的性能，为此对数据集 Polyvore-53 进行了拆分，拆分后数据集的统计数据如表 4-2 所示，统计了不同拆分情况下平均每个用户拥有的正负样本数量，其中负样本包括上述两类。

表 4-2 Polyvore-53 拆分数据集的统计信息

	训练集		验证集		测试集	
切分	正样本数	负样本数	正样本数	负样本数	正样本数	负样本数
100%	200	4000	20	400	40	800
10%	20	400	20	400	40	800
5%	10	200	20	400	40	800
5%*	10	20	20	400	40	800

4.3 评估指标

对于服装推荐任务我们对两个推荐任务进行了实验。

一是服装推荐，即对于每个用户，我们按照其兼容性得分的降序对测试服装进行排名。排名性能通过接收者操作特征曲线下的面积（Area Under the receiver operating characteristic Curve，AUC）和归一化贴现累积收益（Normalized Discounted Cumulative Gain，NDCG）进行评估。

二是填空（fill-in-the-blank，FITB）时尚推荐实验。我们的目标是从一组候选项目中选择一个项目（在我们的实验中有四个），该项目与服装的其余项目最兼容。最基本的事实项目是正确的答案，表现是通过答案的准确性来衡量的。

4.4 算法

4.4.1 超参数β

我测试了超参数$\boldsymbol{\beta}$取不同值的情况下推荐系统的性能表现如下图。其中标量$\boldsymbol{\beta}$用于式（15）中平衡用户编码中基于中用户相似度编码和基于物品相似度编码的权重。$\boldsymbol{\beta}$的取值范围应为[0, 1]。

如实验结果所示，性能随着$\boldsymbol{\beta}$从小变大先上升再下降，当$\boldsymbol{\beta}$=0.5 时，AUC 指标最佳，而当$\boldsymbol{\beta}$=0.3 时，NDCG 指标最佳。于是最终我选择$\boldsymbol{\beta}$=0.5。

表 4-3 β 不同取值下评估的 AUC，NDCG

beta	AUC	NDCG
0.1	0.8053	0.6526
0.3	0.8057	0.6593
0.4	0.8053	0.6579
0.5	0.8064	0.6564
0.6	0.8054	0.6536
0.7	0.8047	0.6508
0.9	0.7981	0.6359

4.4.2 超参数ϵ

我测试了超参数$\boldsymbol{\epsilon}$取不同值的情况下推荐系统的性能表现如下图。其中标量$\boldsymbol{\epsilon}$用于式（17）中平衡用户相似度中正样本和负样本的权重。$\boldsymbol{\epsilon}$的取值范围应为[0, 1]。

如实验结果所示，性能随着$\boldsymbol{\epsilon}$从小变大先上升再下降，当$\boldsymbol{\epsilon}$=0.5 时，AUC 指标最佳，而当$\boldsymbol{\epsilon}$=0.7 时，NDCG 指标最佳。于是最终我选择$\boldsymbol{\epsilon}$=0.5。

表 4-4 ε 不同取值下评估的 AUC，NDCG

epsilon	AUC	NDCG
0.3	0.8045	0.6526
0.5	0.8064	0.6564
0.6	0.8057	0.6560
0.7	0.8053	0.6583
1.0	0.8048	0.6567

4.4.3 算法性能

图 4-1 展现了不同算法的性能差别。在 4.2.2 章节中提出的四个不同大小的数据集上做了评估，同时都使用了 Polyvore-630 预训练。

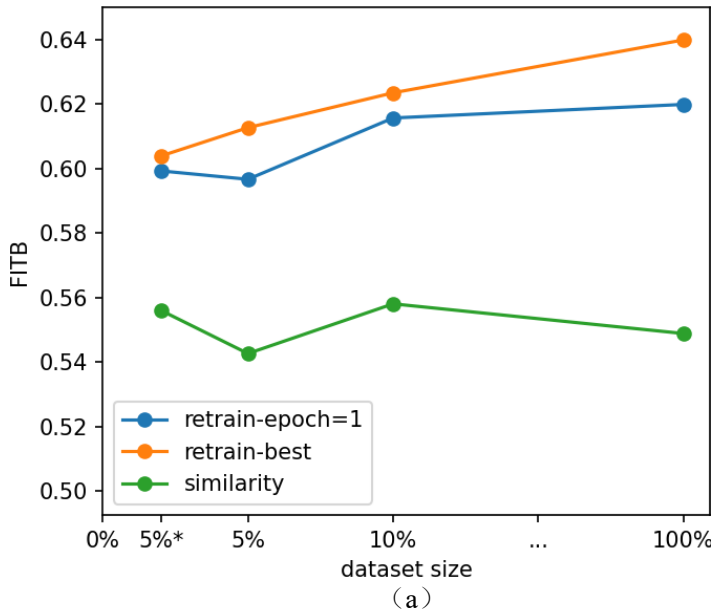
共比较了三种不同的算法：

(1) retrain-epoch=1：重新训练神经网络中用户编码部分，训练 epoch=1。

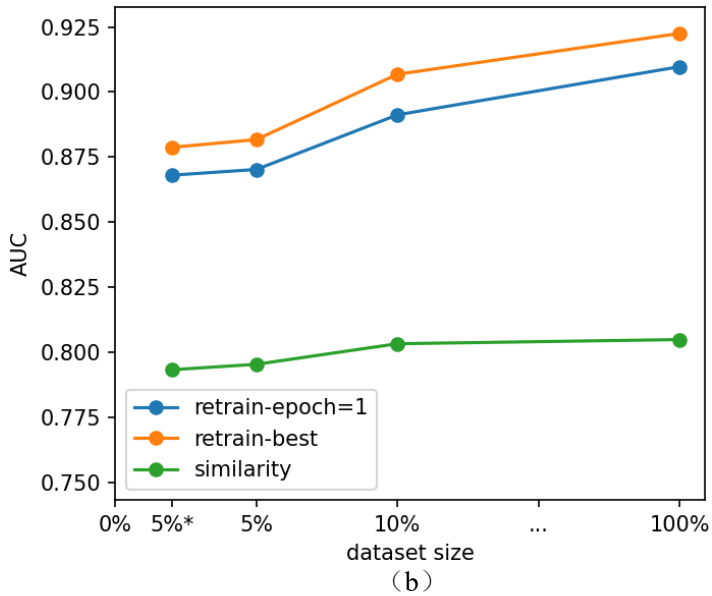
(2) retrain-best: 重新训练神经网络 epoch=20，并取 loss 最小的模型。训练的详细信息见表 4-5。

(3) similarity：使用了 2.4 章节提出的基于相似度的算法。

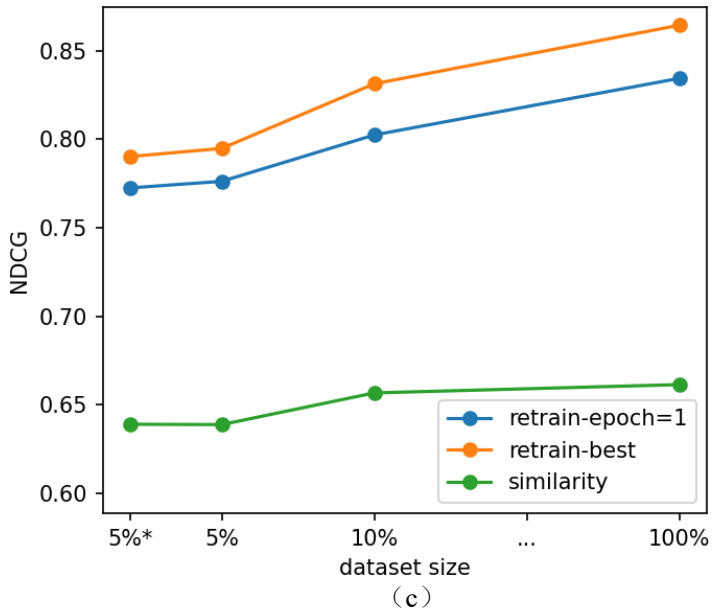
三种模型分别在 4.2.2 章节提出的两种任务中进行测试，三张图分别表示了三种指标的数值随着数据集大小改变而变化。因此可以得出以下结论：



（a）



（b）



（c）

图 4-1 不同数据集大小下不同算法的性能。（a）（b）（c）分别为 FITB，AUC，NDCG 评估

(1) 基于相似度的算法在性能上略低于基于神经网络的算法，但差距在可接受范围之内。

(2) 随着数据集变小，基于神经网络的算法性能下降速度快，而基于相似度的算法则随着数据集变小性能几乎没有下降，十分稳定。

(3) 训练多个 epoch 能使神经网络算法达到更好的性能，达到最低的 loss 平均需要 10 epoch。

表 4-5 retrain 方法训练数据

数据集切分	总的时期（Total Epoch）	最好的时期（Best Epoch）
100%	20	10
10%	20	13
5%	20	8
5%*	20	10

4.4.4 算法效率

我对比了基于神经网络的算法和基于相似度的算法两者之间的效率差距，其中神经网络需要在显卡上运行，否则速度会降至数小时。基于神经网络的算法训练过程全部需要在线执行，且效率缓慢，而基于相似度的算法需要一定时间离线计算物品的编码信息，而在线计算部分则远快于基于神经网络的算法。注意这里是对 53 个用户进行训练，如果只针对单个用户，基于相似度的算法能够在几百毫秒时间内完成计算，实际运用场景的测试见下文。

表 4-6 两种方法在 Polyvore-53 数据集上训练用时

方法	批大小	线程数	硬件	离线耗时	在线耗时
retrain epoch=1	64	4	NVIDIA TITAN RTX	/	88s
similarity	/	1 ↓ 75%	i7-9800X	40s	6s ↓ 93.18%

4.5 网页

4.5.1 后端效率分析

启动后端程序并模拟用户操作，项目包括：

(1) 数据加载：后端启动后初始化过程中从文件加载套装编码信息所需要的时间。

(2) 套装推荐：用户每次访问网页，后端都要为用户筛选出十个套装，测试套装推荐生成所需要的时间。

(3) 响应交互：用户点击“喜欢”或“不喜欢”按钮，后端保存信息并更新用户编码所花费的时间。

我对每个项目进行了五次测试，并取了五次结果的平均值，具体结果见表 4-7。结果显示在线操作（套装推荐，响应交互）均能在一秒时间内做出响应，如此高的性能足够为用户提供流畅的网页体验。

表 4-7 网页开发各项目平均用时

项目	用时（秒）
数据加载	2.42
套装推荐	0.86
响应交互	0.025

4.5.2 推荐表现

图 4-2 初始化网页内容

现在我们来测试网页系统的推荐表现。首先访问网页，初始化时系统认定我们是一个新用户，没有任何信息。因此此时的用户编码是随机生成的，网页推荐给我们了如图 4-2 所示的十个套装。可以发现目前推荐的套装中有五件黑色上衣四件白色上衣，以及七条黑色系裤子裙子，鞋子也基本全是黑灰色调，而且几乎所有推荐的衣服都没有明显的花纹特征，可以任务目前给用户的默认喜好是黑白朴素的。假

设我们这个用户喜欢色彩图案更加艳丽的套装，于是我们给这十个套装进行了评价，从左到右逐行依次选择了不喜欢，喜欢，喜欢，不喜欢，不喜欢，不喜欢，不喜欢，喜欢，不喜欢，不喜欢。如上进行了十次点击操作后刷新网页，此时网页推荐的套装如图 4-3 所示，已经根据用户的喜好生成了更符合个性化的套装推荐，所展示出来的套装已经和原本单调的风格大相径庭，变成了更鲜艳活泼的风格。其中六件衣服是不为黑白且有花纹的，裤子鞋子部分也推荐了有各式各样的花纹物品。

因此可见，我们的网页推荐系统符合套装推荐系统个性化的要求。以上观察是基于主观评测，客观的数据评估可见上文第 4.4.3 节。

图 4-3 用户交互后网页内容

5 总结与展望

本设计方案达到了任务书的要求：理解套装的表征方法，掌握经典的个性化套装推荐算法；基于经典的推荐算法模型，实现基于 Web 的推荐系统的开发。具体表现为：

(1) 提出了一种新的用户编码算法，该算法基于相似度计算用户的哈希编码，极大得提升了新用户冷启动问题下的效率。

(2) 完成了个性化推荐系统的网页开发，前端提供了套装展示和用户喜好交互等功能，后端保证了实时高效的套装推荐，能随着用户喜好的变更做出快速调整。

另外，由于时间、水平和经验有限，本个性化套装推荐系统在诸多方面仍存在不足之处，在未来有一些值得完善提高的地方，主要有以下几个方面：

(1) 基于相似度的编码算法虽然在速度上有明显优势，但是在性能上和基于神经网络的算法仍然存在不小的差距，如何缩小这一差距将是一个值得研究的问题。

(2) 目前网页后端仅仅使用了基于相似度的编码算法，目的是实现实时响应在线推荐的系统。而该算法在性能上仍不及基于神经网络的算法，未来可以加入基于神经网络的算法作为一个离线算法，只在固定时间或者用户点击信息积累到一定程度的时候离线允许，此举能够提升个性化套装推荐的效果。

(3) 受限于数据集等因素，现在的算法统计了用户对一个套装整体的喜好程度而忽略了对单个物品的喜好程度，如果能分开考虑用户对不同物品的喜好对推荐的效果有一定帮助。

(4) 前端的显示效果仍然有提升空间，受限于数据集提供的图片，目前只是将套装各部分（上衣，裤子，鞋子）三张相同尺寸的图片简单得拼接在一起，这并不能很好体现出现实中服装上身的效果。

(5) 目前的网页系统并没有开发用户登录系统，只能对一个用户进行个性化套装推荐，也没有提供用于个人主页等信息。未来可以加入用户系统，绘制用户画像，有助于提升用户体验。

这次毕业设计对我来说，不仅是一个宝贵的机遇，也是一个巨大的挑战。在这次毕业设计中，我通过实践学到了许多知识，不断锤炼了动手能力。同时，通过实际工程的设计，我深刻认识到了理论知识和实践应用之间的区别。在应用过程中，我遇到了各种问题，需要仔细分析，并逐步解决。这次毕业设计让我在实践中不断成长，提高了解决问题的能力，为我未来的职业发展奠定了坚实的基础。

致谢

在完成本篇论文期间，我收获了许多宝贵的经验和知识，并得到了很多人的支持和帮助。在此，我想向所有帮助我的人致以最真诚的感谢和敬意。

首先，我要感谢我的导师谭老师。谭老师一直以来都非常关心和支持我的学术研究，他的教导和指导对我研究方向的确定和研究方法的选择起到了非常重要的作用。在研究过程中，他不断地鼓励我，激发我对研究的热情，并且给予了我很多有益的建议和指导。谭老师的辛勤工作和专业知识对我的学术发展产生了重要影响，我感到非常幸运和荣幸能够成为他的学生。

此外，我还要感谢母校对我的培养和支持。我在母校度过了四年难忘的学习生涯，学到了很多知识和技能，结识了许多优秀的同学和老师。母校的教育理念和学风对我的成长和发展产生了深远的影响，使我在学术和社会方面都更加成熟和自信。我会继续为母校争光，为祖国贡献自己的力量。

特别的，我非常感激 ChatGPT 和翻译软件，因为它们在我的论文研究中发挥了重要作用。ChatGPT 作为一种大型语言模型，为我的写作提供了很多帮助，使我能够更快速地完成论文，并提供了一些有用的建议和参考。翻译软件也在我的研究中发挥了重要作用，让我能够更容易地了解来自其他语言的重要文献和参考资料。在此，我向 ChatGPT 和翻译软件的开发人员表示最深刻的谢意。感谢你们让我完成了这篇论文，也让我的研究更加丰富和全面。

最后，我要感谢我的家人和朋友。他们的支持和鼓励使我在学术研究和生活中更加坚强和自信。他们的爱和关心是我不断前行的动力和源泉。

在此，再次向所有支持和帮助过我的人表示最真诚的感谢和敬意！

参考文献

[1] Lu Z, Hu Y, Jiang Y, 等. Learning Binary Code for Personalized Fashion Recommendation[J]. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)[C]. Long Beach, CA, USA: IEEE, 2019: 10554–10562.

[2] Chen W, Huang P, Xu J, 等. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion[J]. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining[C]. Anchorage AK USA: ACM, 2019: 2662–2670.

[3] Li X, Wang X, He X, 等. Hierarchical Fashion Graph Network for Personalized Outfit Recommendation[J]. arXiv, 2020.

[4] Lin Y, Moosaei M, Yang H. OutfitNet: Fashion Outfit Recommendation with Attention-Based Multiple Instance Learning[J]. Proceedings of The Web Conference 2020[C]. Taipei Taiwan: ACM, 2020: 77–87.

[5] Zhan H, Lin J, Ak K E, 等. A^3-FKG: Attentive Attribute-Aware Fashion Knowledge Graph for Outfit Preference Prediction[J]. IEEE Transactions on Multimedia, 2022, 24: 819–831.

[6] Liu H, Wang R, Shan S, 等. Deep Supervised Hashing for Fast Image Retrieval[J]. 2016: 2064–2072.

[7] Cao Z, Long M, Wang J, 等. HashNet: Deep Learning to Hash by Continuation[J]. 2017: 5608–5617.

[8] Rendle S, Freudenthaler C, Gantner Z, 等. BPR: Bayesian personalized ranking from implicit feedback[J]. arXiv preprint arXiv:1205.2618, 2012.

[9] Han X, Wu Z, Jiang Y-G, 等. Learning fashion compatibility with bidirectional lstms[J]. Proceedings of the 25th ACM international conference on Multimedia[C]. 2017: 1078–1086.

[10] Vasileva M I, Plummer B A, Dusad K, 等. Learning type-aware embeddings for fashion compatibility[J]. Proceedings of the European conference on computer vision (ECCV)[C]. 2018: 390–405.

[11] Hu Y, Yi X, Davis L S. Collaborative fashion recommendation: A functional tensor factorization approach[J]. Proceedings of the 23rd ACM international conference on Multimedia[C]. 2015: 129–138.

[12] Lu Z, Hu Y, Chen Y, 等. Personalized Outfit Recommendation with Learnable Anchors[J]. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition

(CVPR)[C]. Nashville, TN, USA: IEEE, 2021: 12717–12726.
[13]Guan W, Song X, Zhang H, 等. Bi-directional Heterogeneous Graph Hashing towards Efficient Outfit Recommendation[J]. Proceedings of the 30th ACM International Conference on Multimedia[C]. Lisboa Portugal: ACM, 2022: 268–276.
[14]Zhan H, Lin J. PAN: Personalized Attention Network For Outfit Recommendation[J]. 2021 IEEE International Conference on Image Processing (ICIP)[C]. Anchorage, AK, USA: IEEE, 2021: 2663–2667.

附录

在这里详细描述本文所提出的基于相似度的用户编码算法推导过程。

记套装数量为L, 记单品类数为N, 记特征维度为D, 用户数量U。

定义$\boldsymbol{b}_u$为算法所求的用户哈希编码：

$$\boldsymbol{b}_u = \beta \cdot \boldsymbol{b}_{u-user} + (1-\beta) \cdot \boldsymbol{b}_{u-item}.$$

其由$\boldsymbol{b}_{u-user}, \boldsymbol{b}_{u-item}$两部分组成，分别表示基于用户相似度的编码以及基于物品相似度的编码，标量β用于平衡这两个项的贡献。

定义$s_{O_u^{posi}, O_{u'}^{posi}}$为两个用户喜欢的套装之间的相似度：

$$s_{O_u^{posi}, O_{u'}^{posi}} = \frac{1}{L_u^{posi} L_{u'}^{posi}} \sum_i^{L_u^{posi}} \sum_j^{L_{u'}^{posi}} \frac{1}{ND} \sum_n^N \boldsymbol{b}_{u_i^{posi}}^{(n)\top} \boldsymbol{\Lambda}^{(i)} \boldsymbol{b}_{u_j'^{posi}}^{(n)},$$

其中$\mathcal{O}_u^{posi}$表示用户u喜欢的所有套装列表，反之$\mathcal{O}_u^{nega}$表示用户u不喜欢的所有套装列表。同理可得$s_{O_u^{nega}, O_{u'}^{posi}}, s_{O_u^{posi}, O_{u'}^{nega}}$，不再赘述。

定义$r_{u,u'}$表示两个用户的相似度

$$r_{u,u'} = \frac{r_{O_u^{posi}, O_{u'}^{posi}} - \epsilon (r_{O_u^{nega}, O_{u'}^{posi}} + r_{O_{u'}^{nega}, O_u^{posi}})}{1+2\epsilon},$$

其中标量ϵ用于平衡正负样本的贡献，$0 < \epsilon < 1$。

于是我们得到：

$$\boldsymbol{b}_{u-user} = \frac{1}{U} \sum_i^U r_{u,u_i} \boldsymbol{b}_u^{(i)}.$$

此时基于用户相似度编码计算的时间复杂度为$O(UL^2ND)$，我们不难发现该计算的瓶颈在于套装之间的相似度$s_{O_u, O_{u'}}$的计算，其时间复杂度为$O(L^2ND)$。根据乘法集合率，我们不难将式子转化成：

$$s_{O_u^{posi}, O_{u'}^{posi}} = \frac{1}{ND} \left(\frac{1}{L_u^{posi}} \sum_i^{L_u^{posi}} \sum_n^N \boldsymbol{b}_{u_i^{posi}}^{(n)} \right)^\top \boldsymbol{\Lambda}^{(i)} \left(\frac{1}{L_{u'}^{posi}} \sum_j^{L_{u'}^{posi}} \sum_n^N \boldsymbol{b}_{u_j'^{posi}}^{(n)} \right),$$

此时我们记$\boldsymbol{b}_{u^{posi}}^{mean}$为用户$u$喜欢的所有套装的编码平均值：

$$\boldsymbol{b}_{u^{posi}}^{mean} = \frac{1}{L_u^{posi}} \sum_j^{L_u^{posi}} \sum_n^N \boldsymbol{b}_{u_j^{posi}}^{(n)}.$$

同理可得用户u不喜欢的所有套装的编码平均值$\boldsymbol{b}_{u^{nega}}^{mean}$此处不在赘述。此时均值计算的时间复杂度为$O(LND)$，而基于用户相似度的编码时间复杂度为$O(UD)$。

求解基于物品相似度编码时需要求解以下式子：

$$\underbrace{\arg\max}_{\boldsymbol{b}_{u-item}} r_{u,O^{posi}} = \frac{1}{L^{posi}} \sum_i^{L^{posi}} r_{u,O_i^{posi}}, \text{ and}$$

$$\underbrace{\arg\min}_{\boldsymbol{b}_{u-item}} r_{u,O^{nega}} = \frac{1}{L^{nega}} \sum_i^{L^{nega}} r_{u,O_i^{nega}} \quad ,$$

其等价于：

$$\underbrace{\arg\max}_{\boldsymbol{b}_{u-item}} \frac{r_{u,O^{posi}}-r_{u,O^{nega}}}{2} = \frac{1}{2DN}\left(\frac{1}{L^{posi}}\sum_{i}^{L^{posi}}\sum_{n}^{N}\boldsymbol{b}_{i_n^{posi}}^{(n)} - \frac{1}{L^{nega}}\sum_{i}^{L^{nega}}\sum_{n}^{N}\boldsymbol{b}_{i_n^{nega}}^{(n)}\right)^{\top}\boldsymbol{\Lambda}^{(u)}\boldsymbol{b}_{u-item}$$

，

因为用户和物品的哈希编码都是二进制编码$b_i, b_u \in \{-1,+1\}^D$，因此要使得式子的值最大，可以看作求解两个D维向量点积的最大值。当点积达到最大值时，两个向量平行。如果加入$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1,\ldots,\lambda_D)$，我们观察发现只有其值的正负会影响最终结果，因此得出以下解：

$$\boldsymbol{b}_{u-item} = \frac{1}{2DN}\cdot \mathrm{sign}(\boldsymbol{\Lambda}^{(u)})\left(\frac{1}{L^{posi}}\sum_{i}^{L^{posi}}\sum_{n}^{N}\boldsymbol{b}_{i_n^{posi}}^{(n)} - \frac{1}{L^{nega}}\sum_{i}^{L^{nega}}\sum_{n}^{N}\boldsymbol{b}_{i_n^{nega}}^{(n)}\right),$$

此时基于物品相似度编码计算的时间复杂度为$O(LND)$。不难看出该式子也可以表示为：

$$\boldsymbol{b}_{u-item} = \mathrm{sign}(\boldsymbol{\Lambda}^{(u)})(\boldsymbol{b}_{u^{posi}}^{mean} - \boldsymbol{b}_{u^{nega}}^{mean}).$$

此时均值计算的时间复杂度为$O(LND)$，而基于物品相似度的编码时间复杂度为$O(D)$。

综上所述离线计算的时间复杂度$O(LND)$，在线计算时间复杂度为$O(UD)$。