LONDON'S GLOBAL UNIVERSITY
Department of Computer Science

Large Language Models for Product Substitution

Zhiqi Huang

Academic Supervisor: Professor Philip Treleaven

Industry Supervisors (Tesco): Ms Loucine Tovmassian,

Ms Ward Dib and Mr Miguel Cano Puerto

University College London

A Project Report Presented in Partial Fulfillment of the Degree

Master of Science

September 2024

Abstract

This thesis is an original research exploring how Large Language Models (LLMs) can improve product substitution in offline scenarios. A task that occurs when a requested product is unavailable and must be replaced with a similar alternative. This is an empirical study into whether incorporating LLMs into recommendation systems can outperform traditional methods in the product substitution task. This study was conducted in collaboration with the Tesco data science team using anonymous offline acceptance and online click-stream data.

This thesis aims to integrate recent advances in Natural Language Processing (NLP) to develop a recommendation system that addresses areas where traditional models struggle. Conventional methods, which rely heavily on historical data and basic semantic features, fail to reason about the complex relationships between products and encounter problems such as the cold-start problem. In addition, user behaviour data, while useful, can be biased and unreliable in representing product substitution relationships. To overcome these limitations, it is essential to develop recommendation systems that consider user behaviour, fully interpret product information, and have access to open-world, up-to-date knowledge.

The proposed recommendation system uses advances in NLP to accurately predict outcomes using only product text information provided by the retailer. Performance can be further improved by fine-tuning on user behaviour data. Our dual-component recommendation system, which combines an LLM with Retrieval-Augmented Generation (RAG) and a fine-tuned cross-encoder, has shown promising results, particularly in categories with complex product information. Specifically, it achieved increases in NDCG scores, ranging from 0.04% to 0.73%, and in NDCG_f scores, ranging from 6.86% to 32.51%, compared to the naive baseline.

The thesis consists of three investigations, which the author believes are original contributions to the application of LLMs to the product substitution task. The structure of the research is presented below:

1. **Datasets:** This section provides a detailed overview of the datasets used in the project, covering data sources, pre-processing methods and analysis techniques. It provides insights into key features of the datasets, such as product division distribution, substitute count distribution and Bayesian acceptance rate distribution, which are valuable in developing a recommendation system tailored to the datasets.

2. **Methodology Design:** This section outlines the design of the recommendation system, explaining the rationale behind each component, along with detailed descriptions and key mathematical concepts. The chapter concludes with implementation details, including an overview of the training and validation processes.

3. **Experiments:** This section presents experiments designed to optimise and evaluate the performance of the recommendation system. It introduces the experimental design, evaluation metrics, and baseline models, followed by three experiments that examine the effects of LLM prompt styles, product information inputs, and fine-tuning and hyper-parameters on the performance of the system.

The thesis presents the following original contributions to science:

• **Reframed the Substitution Task as an NLP Problem**: This thesis reframes product substitution as a natural language processing problem in order to exploit the interpretability and generative capabilities of language models.

• **Proposed a New Recommendation System Integrating LLMs with Cross-Encoder**: A novel recommendation system was proposed and implemented that combines LLM with RAG and cross-encoder models, balancing accurate domain-specific product representation and scalability.

• **Conducted Extensive Experiments and Ablation Studies on TESCO Datasets**: Extensive experiments and ablation studies were conducted on the Tesco datasets, demonstrating improved performance over the baseline models across various product categories. This effectively improves performance across different product categories and demonstrates the generalisability of the method.

Keywords— Large Language Model - Recommendation System - Product Substitution

Impact Statement

This study explores the development and implementation of AI-driven recommendation systems to provide ranked substitute product suggestions considering semantic similarity, price considerations, and product popularity. This research has significant implications for the retail industry and the general public:

- **Impact on Retail:** By providing relevant product substitution suggestions, the AI-driven system can significantly improve customer satisfaction for Tesco in the UK. This improved product substitution can help the retailer reduce waste and costs, leading to more efficient supply chain management and better resource utilisation.
- **Impact on Broader Industrial Applications:** The system can be adapted for car dealerships, clothing retailers, and electronics retailers to recommend alternative products. In car dealerships, it can suggest vehicle models to increase sales and satisfaction. Clothing retailers can use it to recommend clothes based on style, size, and price, creating a better shopping experience. Electronics retailers can use it to suggest gadgets or appliances based on specifications and price, improving sales efficiency.
- **Impact on the Public:** The AI-driven recommendation model promotes sustainable consumption by guiding consumers towards available substitutes, reducing the probability of panic buying and overstocking due to fear of shortages. This balanced approach helps to maintain an equilibrium between supply and demand. In the context of public health, the model can be used by health organisations to suggest healthier food alternatives, encouraging better dietary choices and potentially improving public health outcomes.

Declaration

I, Name, I declare that the thesis has been composed by myself and that the work has not be submitted for any other degree or professional qualification. I confirm that the work submitted is my own, except where work which has formed part of jointly-authored publications has been included and referenced. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Zhiqi Huang	09/09/2024
Signature	*Date*

Table of Contents

List of Figures vii

List of Tables ix

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Research Objectives . . . 3
- 1.3 Research Experiments . . . 4
- 1.4 Scientific Contribution . . . 4
- 1.5 Thesis Structure . . . 5

2 Background and Literature Review 7
- 2.1 Background on Large Language Models . . . 7
  - 2.1.1 Evolution of Large Language Models . . . 7
- 2.2 Background on Recommendation Systems . . . 12
  - 2.2.1 Traditional Recommendation Systems . . . 12
  - 2.2.2 LLM-Based Recommendation . . . 14
- 2.3 Background on Identifying Substitute Products . . . 20
- 2.4 Chapter Summary . . . 21

3 Datasets 22
- 3.1 Datasets . . . 22
  - 3.1.1 Datasets Description . . . 22
  - 3.1.2 Datasets Pre-processing . . . 24
  - 3.1.3 Datasets Analysis . . . 25
- 3.2 Chapter Summary . . . 30

4 Methodology Design 31

4.1 LLM with RAG as Feature Augmenter . . . 32
4.2 Cross-Encoder as Candidate Re-ranker . . . 33
4.2.1 Cross-Encoder Architecture . . . 34
4.2.2 Cross-Encoder Implementation Details . . . 40
4.3 Chapter Summary . . . 41

5 Experiments **42**
5.1 Overview . . . 42
5.1.1 Evaluation Metrics . . . 43
5.1.2 Baselines . . . 45
5.2 Experiment 1: Prompting Style . . . 46
5.2.1 Task Objectives and Description . . . 46
5.2.2 Results . . . 46
5.3 Experiment 2: Input to Cross-Encoder . . . 50
5.3.1 Task Objectives and Description . . . 50
5.3.2 Results . . . 51
5.3.3 Ablation Study: Finetuning Effect . . . 56
5.4 Experiment 3: Hyper-parameter tuning . . . 57
5.4.1 Task Objectives and Description . . . 57
5.4.2 Results . . . 57
5.5 Chapter Summary . . . 58

6 Conclusion and Future Work **60**
6.1 Conclusion . . . 60
6.2 Future Work . . . 61

References **63**

Appendix A Further Details of Results **70**
A.1 Experiment 2 Results . . . 70
A.1.1 More Detailed Breakdown of Results on Different Cross-Encoder Inputs . . . 70
A.1.2 More Detailed Breakdown of Results on Finetuning Effect . . . 71
A.2 Experiment 3 Results . . . 72
A.2.1 More Detailed Breakdown of Results on Hyperparamter Tuning . . . 72

List of Figures

2.1 Structure of RNN . . . 8
2.2 Structure of LSTM . . . 9
2.3 Illustration of MLM Task for BERT Pretraining . . . 10
2.4 Illustration of Embeddings used in NSP Task for BERT Pretraining . . . 11
2.5 Overview of LLMs in RS . . . 15
2.6 Researches of Integrating LLMs into Different Stages of RS pipeline . . . 17
2.7 Trend of Development of LLM-Based Recommendation System . . . 18

3.1 Distribution of Substitute Counts for Online, Offline, and Candidate Data . . . 27
3.2 Bayesian Acceptance Rate Updates . . . 29
3.3 Bayesian Acceptance Rate Distribution in Tesco-Hybrid Across Different Divisions 30

4.1 Overview of Pipeline . . . 31
4.2 Overview of Cross-Encoder . . . 34
4.3 Tokenizer . . . 35
4.4 Embedding Module . . . 37
4.5 Encoder Module . . . 39
4.6 Pooling Layer . . . 40

5.1 Illustration of Substitution Process and Low Compliance Rate . . . 44
5.2 Example Prompts for Division W . . . 47
5.3 Comparison of Model Results . . . 48
5.4 Examples for Comparing Different Prompting Styles . . . 49
5.5 Input Combinations Tested for Cross-Encoder Training . . . 50
5.6 Comparison between Baselines and Cross-Encoders for Division F, T and B . . . 51
5.7 Comparison between Baselines and Cross-Encoders for Division W and N . . . 52
5.8 Comparison between Product Title and Summarized Descriptions from LLM . . . 55
5.9 Heat Map: Ablation Study Results on Fine-tuning Effect . . . 56

5.10 Heat Maps for NDCG and NDCG$_f$ Scores For Different Hyperparameters By Division 59

A.1 Results for Hyper-parameter Tuning 72

List of Tables

2.1 An Example of a Utility Matrix 13

3.1 Tesco-Product Dataset Fields 23
3.2 Tesco-Online Dataset: Mock Data 23
3.3 Tesco-Offline Dataset: Mock Data 24
3.4 Tesco-Candidate Dataset: Mock Data 24
3.5 Division Statistics 26

5.1 Size of Training Set for Each Division 53
5.2 Parameters for Each Division that Achieve Highest NDCG Score 58

A.1 Results of Experiment 2 (Different Inputs to Cross-Encoders) 70
A.2 Ablation Study Results on Finetuning Effect 71

List of Abbreviations

BCE: Binary Cross-Entropy

BERT: Bidirectional Encoder Representations from Transformers

CRM: Conventional Recommendation Models

CTR: Click-Through Rate

ELU: Exponential Linear Unit

GPT: Generative Pre-trained Transformer

GeLU: Gaussian Error Linear Unit

LDA: Latent Dirichlet Allocation

LLM: Large Language Model

LSTM: Long Short-Term Memory

LoRA: Low-Rank Adaptation

NLP: Natural Language Processing

PEFT: Parameter-Efficient Fine-Tuning

PLM: Pretrained Language Models

RAG: Retrieval-Augmented Generation

RNN: Recurrent Neural Network

RS: Recommendation System

ReLU: Rectified Linear Unit

Chapter 1

Introduction

This chapter begins by discussing the motivations behind the project, emphasizing the real-world problem the research aims to address and the rationale for incorporating NLP techniques into a recommendation system. This is followed by a clear presentation of the research objectives and a brief overview of the research experiments. The chapter also states the scientific contributions of the thesis, highlighting its potential impact on the field. Finally, it provides the structure of the thesis, guiding the reader through the organization and flow of the subsequent chapters.

Large language models are advanced artificial intelligence program designed to understand and generate human-like text. They are termed "large" because they are trained using extensive data and require massive computational needs, and "language models" for their ability to model probability of word sequences.

LLMs' development began with rule-based systems in the 1950s. In the 1990s and early 2000s, statistical and machine learning based models using probabilities to predict words gained popularity. From the mid-2000s to the 2010s, deep learning techniques like Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) improved context understanding by remembering longer information. In 2017, the introduction of the Transformer architecture revolutionized LLMs, leading to advanced models like OpenAI's GPT, Google's BERT, Meta's LLaMA, and Mistral AI, which are used in this research. Today, LLMs are widely used in different fields from robotics to e-commerce to finance. In this research, we will be focusing on the application of LLMs in e-commerce.

In e-commerce, substitutions are any point in the customer journey where one item is replaced with another. Online substitutions occur when an item is either derraged or unavailable on website and offline substitutions occur when an item is ordered on the website but become unavailable at picking time. In this research, we will be focusing on offline substitutions.

1.1 Motivation

The key motivations for improving recommendation systems in product substitutions are multi-faceted, with aims for enhancing both customer experience and business performance. From the aspect of customer experience, providing accurate and relevant product substitutions enables customers to receive suitable alternatives when their preferred items are unavailable, enhancing their shopping experience and satisfaction. From the perspective of business performance, effective substitution recommendations help manage stock levels and maintain sales volume efficiently by directing customers to products that are in stock, reducing the probability of lost sales due to out-of-stock items. From the perspective of competitors, the highly homogeneous nature of grocery shopping means that Tesco faces competition from other major companies in the sector, including Sainsbury's, Asda, and Morrisons. Ensuring that customers can always receive satisfactory alternatives is crucial for maintaining customer loyalty and retention, as customers are less likely to turn to competitors when their desired products are unavailable. Advanced recommendation systems can differentiate Tesco from its competitors by offering a more seamless and enjoyable shopping experience, which in turn can attract more customers.

Current recommendation systems for product substitutions heavily depend on two approaches: graph-based and statistical methods. However, these methods overly rely on users' historical behavioral data. Although this data can indirectly suggest substitution relationships between products, it is often biased and should not be used as a direct indicator of substitution relationships [1]. This highlights the need for recommendation systems that also consider the retailer's perspective. Moreover, an over-reliance on biased historical data results in recommending irrelevant products, leading to the low compliance rate issue currently faced by Tesco's Data Science Team. This problem arises when offline store pickers reject model recommendations and spend additional time searching for better substitutes, causing inefficiencies.

By incorporating NLP techniques, our goal is to develop a novel recommendation system that overcomes current limitations and delivers improvements that can not be achieved by existing solutions. From my research, this is the first research that approaches the product substitution task through a purely NLP approach. This is because previous NLP methods such as Word2Vec or TF-IDF, which rely on the presence and frequency of words in the training corpus, may not effectively capture the underlying meanings and complex characteristics of products. This limitation arises due to products with unclear or inconsistent naming conventions and consumers with dynamic preferences.

The development of pre-trained language models in recent years has significantly transformed the approach to product recommendations. Advanced models like BERT [2], GPT [3, 4], and

Llama [5] are trained on large datasets that encompass a broad range of knowledge domains. This extensive training equips them with open-world knowledge and common sense, enabling them to understand the key functionalities of products more effectively. As they can understand context, nuance and the relationships between words in a way that previous models could not. However, while these pre-trained models possess general knowledge, they may not perform optimally on tasks requiring specialized understanding. Fine-tuning enables pre-trained models to adapt to specific tasks, improving performance while being more time- and cost-efficient than training from scratch.

We will also leverage the cross-encoder architecture, which builds on the transformer architecture. It has revolutionised the comparison of semantic similarity between sentences. Cross-encoders directly compare pairs of sentences, allowing for a more detailed and accurate assessment of semantic similarity. This capability is particularly beneficial when comparing product descriptions, as it enables a deeper understanding of the similarities and differences between products based on their descriptions. This leads to more accurate and relevant product substitutions, improving the overall user experience on e-commerce platforms.

1.2 Research Objectives

- The first objective of this research is to conduct a detailed analysis of product data, user online click-stream data, and offline acceptance data. By thoroughly examining these datasets, we aim to identify patterns in the distribution of product category, substitution counts and acceptance rates. The insights gained from this analysis will help guide the design of the product substitution recommendation system.

- The second objective of this research is to develop a product substitution recommendation system that combines an LLM and a cross-encoder. This automated pipeline will handle data retrieval, data pre-processing, model training, and generating a ranked list of recommendations. The LLM will be used for feature augmentation, while the cross-encoder will compare and rank substitutes. The aim is to create a scalable system that seamlessly incorporates advanced language models to improve substitution recommendations.

- The third key objective is to test whether this system can outperform industry baseline methods in addressing the product substitution problem. We will also conduct experiments to examine how different prompting styles, types of product information, and hyperparameter values affect the system's performance, in order to provide a comprehensive analysis of the system.

1.3 Research Experiments

1. **Impact of Prompting Styles on LLM Product Representation Performance:** This experiment examines how different prompting strategies influence the product representation performance of LLMs. By implementing and comparing two distinct prompting strategies–direct summarization and feature extraction–the objective is to determine which approach more accurately captures product features, leading to more precise and relevant recommendations.

2. **Effect of Input Product Information on Cross-Encoder Performance:** This experiment investigates whether fine-tuning the cross-encoder model with LLM-augmented product information can improve performance. To provide a comprehensive analysis, we evaluate four different configurations and compare their performance against two baselines. Additionally, we conduct ablation studies to determine the specific performance boost provided by fine-tuning on Tesco datasets, isolating its impact from other factors.

3. **Influence of Hyperparameters on Cross-Encoder Performance:** This experiment focuses on how hyperparameter tuning affects the cross-encoder's performance. It involves testing four sets of learning rate and batch size configurations to identify the most effective setups across five product categories and three test weeks.

1.4 Scientific Contribution

This thesis aims to make scientific contributions in the following ways:

- **Reframed the Substitution Task as an NLP Problem**: Substitution tasks in e-commerce are challenging due to the dynamic nature of products and customer preferences. This research addresses the issue by developing a pipeline that uses language models for substitution recommendations. The key innovation is reframing the task as an NLP problem, where substitution recommendations are made by comparing product textual information. This approach aims to provide more accurate solutions, especially when product information is complex or historical user data is limited or biased.

- **Proposed a New Recommendation System Integrating LLMs with Cross-Encoder**: This study proposes and implements an innovative pipeline that integrates an LLM with a cross-encoder for substitution recommendation tasks. By using an LLM with RAG, the pipeline fully leverages the LLM's ability to generate up-to-date, domain-specific product

representations. Incorporating a cross-encoder allows the pipeline to rank a large number of candidate pairs, addressing the scalability limitations typically associated with LLMs. This integration results in a balanced approach that provides accurate recommendations while maintaining scalability.

- **Conducted Extensive Experiments and Ablation Studies on TESCO Datasets**: This research conducted extensive experiments and ablation studies using the TESCO datasets. The results show that the proposed methodology effectively learns the semantic and nuanced relationships between products, outperforming current baseline model in product categories with complex product information.

1.5 Thesis Structure

The structure of this thesis is organized as follows:

- **Chapter 2 - Background and Literature Review:** This chapter offers a comprehensive overview of LLMs, recommendation systems, and their integration. It also reviews various methods for identifying substitutes, including graph-based and probabilistic approaches. This chapter establishes a foundational understanding of this hybrid research area.
- **Chapter 3 - Dataset:** This chapter details the datasets used and outlines the preprocessing methods applied. It establishes a critical foundation for the subsequent methodological design by focusing on the analysis of product categories, substitute count distribution, and acceptance rate distribution.
- **Chapter 4 - Model and Pipeline Design:** This chapter outlines the design of a robust pipeline that integrates an LLM for feature augmentation and a cross-encoder for candidate re-ranking. It discusses the rationale behind the pipeline's design, the selection of these specific models, and provides an in-depth explanation of each component of the cross-encoder. The chapter concludes with implementation details, including an overview of the training and validation process.
- **Chapter 5 - Results:** This chapter presents a series of experiments aimed at optimizing and evaluating the performance of the developed pipeline. It begins with a discussion of the experimental design, evaluation metrics, and baseline models, and then delves into three experiments that explore different aspects of the pipeline: LLM prompting styles, the

effects of input product information, and the impact of fine-tuning and hyperparameters on cross-encoder performance.

- **Chapter 6 - Conclusion and Future Work:** This final chapter summarizes the research findings and discusses how the pipeline successfully met the research objectives. It also addresses the current limitations of the pipeline and outlines potential directions for future work to further improve its performance.

Chapter 2

Background and Literature Review

This chapter provides a comprehensive background and detailed literature review on LLMs and recommendation systems, setting the stage for understanding their potential for identifying substitute products. It first traces the evolution of LLMs from n-gram models and complex neural networks such as RNNs and LSTMs to transformer models and advanced pre-trained language models. The chapter then focuses on recommendation systems, exploring different methods such as content-based, collaborative and hybrid filtering. It also examines how LLMs have been integrated into these systems, detailing the stages and trends. The chapter concludes by reviewing traditional methods for identifying substitutes, providing the basis for later discussions on how LLMs improve the accuracy of recommendation systems in product substitution tasks.

2.1 Background on Large Language Models

Large language models are a type of artificial intelligence model designed to understand and generate human-like text based on input. LLMs are typically based on deep learning architectures, in particular the transformer architecture introduced by [6]. They are trained on large datasets, often containing billions of words from books, articles, websites and other text sources. This training enables LLMs to perform exceptionally well in different natural language processing tasks such as text generation [7], natural language understanding [8] and translation [9].

2.1.1 Evolution of Large Language Models

Early Language Models: N-gram model

Initially, language models relied on statistical methods such as n-gram models, in which the prediction of a token x_i depends only on the last $n-1$ characters $x_{i-(n-1):i-1}$ rather than on the whole history:

$$p\left(x_i \mid x_{1:i-1}\right) = p\left(x_i \mid x_{i-(n-1):i-1}\right)$$

Shannon first described a method similar to n-gram models in his work on information theory [10] in the 1950s, where he had participants repeatedly predict the next letter in a sequence based on the previous text. This experiment helped to show how predictable text can be, and formed the basis for early models of language prediction.

These models were computationally efficient and could be trained on large corpora but struggled with long-range dependencies due to their limited context windows.

Neural Language Models: Advancements with RNNs and LSTMs

The introduction of neural networks in language modeling marked a significant step forward. Pioneered by [11] in 2003, neural language models used neural networks to estimate the probability distribution of the next word in a sequence, allowing for longer context windows compared to n-grams. However, these early neural models were computationally expensive and difficult to scale.

Recurrent Neural Networks (RNNs) [12], including Long Short-Term Memory (LSTM) networks [13], have been designed to process sequences by leveraging their ability to maintain an internal state or memory. RNNs, illustrated in Figure 2.1 [14], consist of layers of neurons, where the output from each layer can be fed as input to the next timestep, forming loops within the network. This architecture allows the network to store information in its memory over time. At each time step, the RNN receives two inputs: the input data for the current time step and the output from the previous time step. The network updates its hidden state based on these inputs, allowing it to maintain a form of memory. This ability allows the RNN to consider both the current input and the previous sequence, allowing it to understand context and temporal relationships within the data.

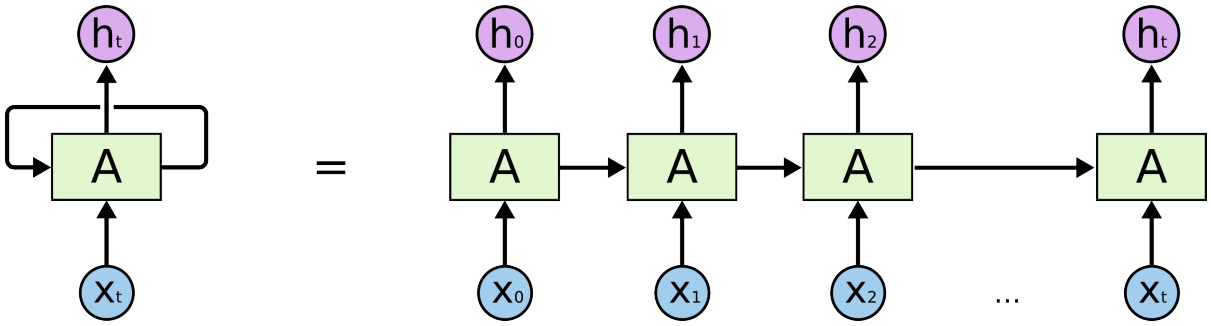


Figure 2.1: Structure of RNN

An LSTM network, illustrated in 2.2, is composed of memory blocks known as cells, each of which contains three types of gates: input, output, and forget gates. These gates control the flow

of information by determining whether to allow new input into the cell (input gate), to remove information that is no longer needed (forget gate), or to allow the information to influence the current output (output gate). The cell can hold values for different time intervals and the three gates manage the movement of information in and out of the cell. This design effectively addresses the vanishing gradient problem often encountered in traditional RNNs, and was better at capturing long-range dependencies within text.

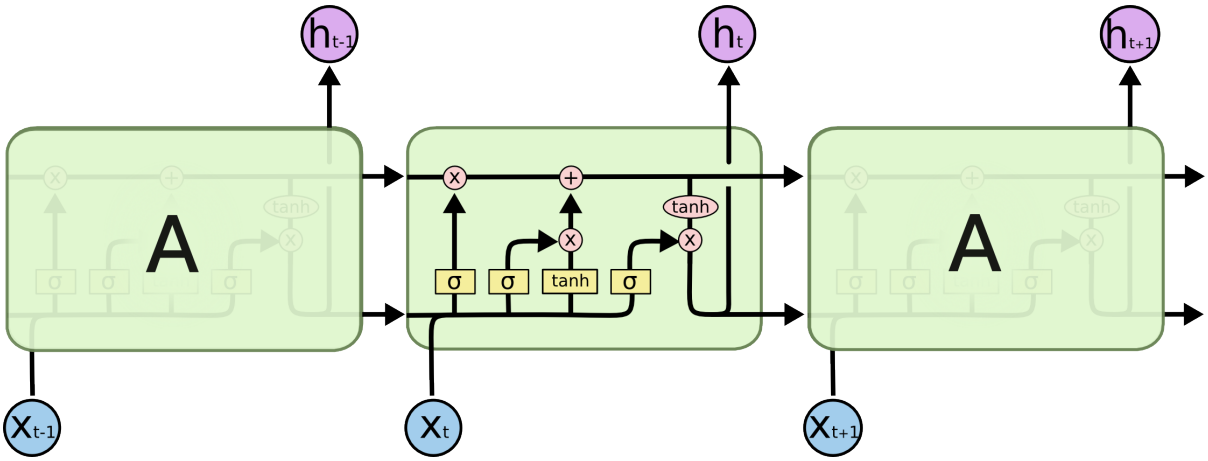


Figure 2.2: Structure of LSTM, as illustrated by Hochreiter and Schmidhuber (1997), and further explained in [14].

Breakthrough with Transformers

The introduction of the Transformer architecture in 2017 by [6] brought a paradigm shift. Transformers are much more effective at processing long sequences and complex structures because they use self-attention mechanisms that weigh the importance of different words in a sequence, regardless of their positional distance. Unlike RNNs, Transformers use a non-recurrent structure that allows inputs to be processed in parallel during training, and in comparison to RNNs that use hidden states, attention weights can be analysed to understand which parts of the input data the model focuses on when making predictions, providing insight into model decisions. This architecture proved to be more effective and efficient for training on large datasets, enabling much larger models to be trained.

Emergence of Pre-trained Language Models

The emergence of pre-trained language models (PLMs) marks a significant advancement in the field of NLP. They are based on the idea of training a model on a large corpus of text data before

fine-tuning it for specific tasks. This pre-training step allows the model to learn a wide-ranging understanding of language, including syntax, semantics, and context.

We will provide in-depth details of the PLM that is used in this project, BERT (Bidirectional Encoder Representations from Transformers). It was introduced by Google in 2018 [2], and was revolutionary for its bidirectional training of transformers. Unlike previous models that processed text in one direction, BERT considers the full context of a word by reading the text from both left to right and right to left, making it highly effective for tasks such as sentence classification and question answering.

Tokenisation is the first component of BERT and involves two main tasks: breaking down raw text into smaller units (called tokens), such as words, subwords or characters, and converting these units into a numerical format that computers can easily process. BERT's tokeniser uses the WordPiece subword tokenisation algorithm. Although the specific training process for WordPiece is not publicly available, its main function is to build a vocabulary of tokens up to a predefined size. This is done by iteratively selecting and merging the highest-scoring character pairs from the text dataset.

BERT is pretrained on two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM involves randomly masking a percentage of the input tokens and then predicting these masked tokens, as shown in Fig. 2.3 [15]. This method is crucial because standard conditional language models are trained to predict text in a single direction, either left to right or right to left. Using bi-directional conditioning, where the model could potentially see the whole sentence, would allow it to easily guess missing words from the surrounding context without a deep understanding of the linguistic relationships. In MLM, 15% of the token positions are chosen randomly for prediction. Of these, 80% are replaced by a [MASK] token, 10% by a random token, and 10% remain unchanged. This forces the model to infer the original token based solely on its contextual understanding.

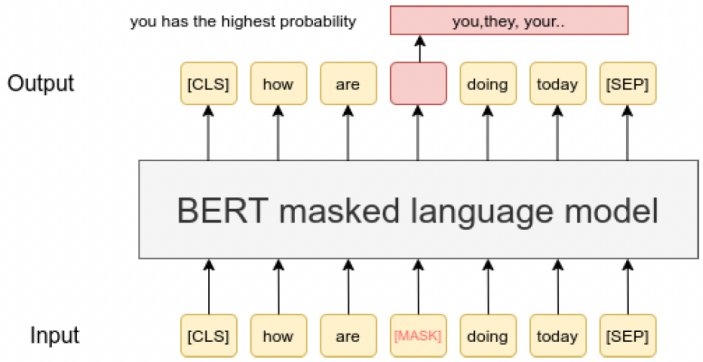


Figure 2.3: Illustration of MLM Task for BERT Pretraining

On the other hand, NSP helps to understand relationships between two sentences, labelled sentence A and sentence B. This task involves predicting whether sentence B logically follows sentence A, thereby improving the model's understanding of narrative flow. In each training example, sentence B is the actual following sentence A 50% of the time (labelled `IsNext`) and a random sentence from the corpus the other 50% of the time (labelled `NotNext`) [2]. Figure 2.4 illustrates the use of three types of embeddings for the NSP task: token embeddings capture semantic meanings, segment embeddings distinguish between sentences A and B, and position embeddings indicate the order of tokens within the sequence.

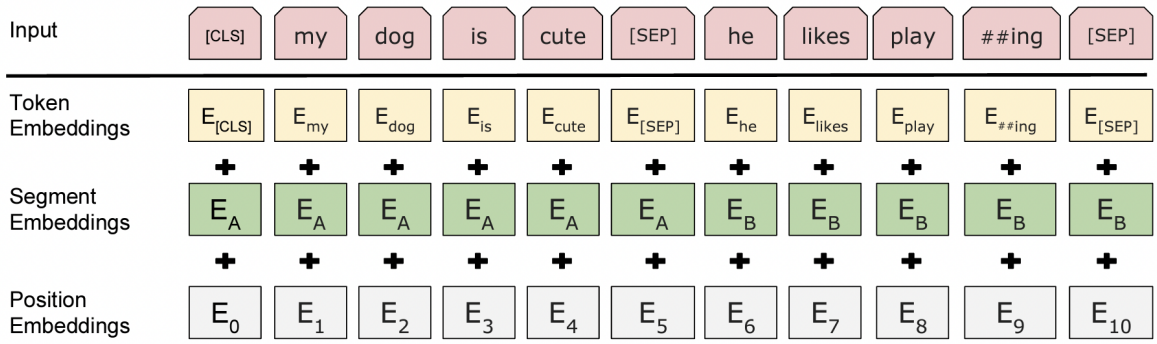


Figure 2.4: Illustration of Embeddings used in NSP Task for BERT Pretraining

Another important PLM is GPT-3, developed by OpenAI [16, 3]. With 175 billion parameters, GPT-3 is much larger and more powerful than previous models, including BERT. GPT-3 is an autoregressive model, which means it generates text by predicting the next word in a sequence based on all previous words. The model is trained to minimise the difference between its predicted words and the actual next words in the training data. The main difference between GPT-3 and BERT lies in their architecture and training objectives. GPT-3's autoregressive nature enables it to generate coherent and contextually relevant text over longer passages, while BERT's bidirectional auto-encoding approach is optimized for understanding the context and relationships within a sentence but is not designed for text generation.

In our project, we will use `gpt-4o-mini`, a smaller variant of GPT-4, which builds on the architecture of GPT-3 but incorporates improvements in architecture and fine-tuning, making it even better at generating coherent text and understanding complex context over long sequences.

2.2 Background on Recommendation Systems

A recommendation system is a type of information filtering system designed to predict the preference or rating that a user would give to an item. These systems are used to suggest items to users based on various criteria, which can include the user's past behaviour, preferences, and similar choices made by other users.

Recommendation systems have been used in many different areas, for example in e-commerce, Amazon [17] and eBay use recommendation systems to suggest products to customers based on previous purchases and browsing behaviour. Netflix [18], Youtube [19] and Spotify [20] use recommendation systems to recommend films, audiobooks and music based on the user's viewing or listening history and what similar users like.

2.2.1 Traditional Recommendation Systems

In this section, we will explore the three main methods used in recommendation systems: content-based, collaborative, and hybrid filtering.

Content-based Filtering

Content-based filtering (CBF) systems make recommendations by analysing the characteristics of items and measuring their similarities based on a user's previous interactions and preferences. We will follow the general formulation for CBF introduced by [21]. Specifically, we consider a training dataset consisting of m item vectors $X = (x_1, \ldots, x_n)$, where each vector has n components. These components can represent binary, nominal or numerical attributes, depending on the use case, and are derived either from the content of the items or from information about the user's preferences. The goal of the learning method is to select a function $f(X)$ based on this training set of m input vectors. This function should be able to classify an unseen item as either positive or negative by returning a binary value or providing a numerical value to represent the degree of similarity.

CBF has an advantage in cold-start problems when encountering new items, as it does not rely on user ratings or interactions from other users. Instead, it uses the inherent characteristics of the items themselves to make recommendations, making it effective even when there is little to no collaborative data available [22]. By using a user's historical interactions, content-based filtering can create a personalised recommendation model that adapts to the user's evolving preferences over time, providing more relevant suggestions [23]. One of the limitations of CBF is that it requires detailed product information to make appropriate recommendations, but this

information is not always available. In addition, CBF tends to focus narrowly on items that are similar to those with which the user has previously interacted, which can lead to a lack of diversity in recommendations. This issue, known as the serendipity problem, means that the model fails to discover new and unexpected items that could be of interest to users [24].

In this project, we will design a content-based recommendation system that uses LLMs to analyse product descriptions and recommend substitutes. While collaborative and hybrid filtering methods are beyond the scope of this project, they are included for a broader understanding of recommendation systems and may inform future work.

Collaborative Filtering

Collaborative filtering predicts a user's interests by analysing the preferences of other users without having to understand the content itself, such as a product description. However, it faces challenges such as cold-start and privacy issues due to shared data. This method relies on user-item interaction data to identify patterns and preferences, and is divided into user-based and item-based [25] approaches. The user-based approach finds similar users and recommends items based on neighbourhood preferences, while the item-based approach predicts new item ratings by calculating the weighted average of ratings from similar items (e.g. if item A is often liked by users who also like item B, then A and B are considered similar).

In collaborative filtering, user-item interaction information is organised into a utility matrix, with users as rows and items as columns (an example is shown in Table 2.1). Entries, either explicit (ratings) or implicit (purchase history), indicate user preferences. The goal is to predict missing matrix entries using historical data. Techniques such as matrix factorization [26] approximate the original matrix with two lower-dimensional matrices representing latent user and item factors. This reduces the dimensionality of the data and addresses issues of sparsity and scalability. Enhancing recommendations with additional features such as demographics or item descriptions can further personalise the system.

	The Matrix	**Avatar**	**Star Wars**	**The Godfather**	**Inception**
User 1	5	3		5	4
User 2	4		5		5
User 3		1	3	2	

Table 2.1: An example of a utility matrix in recommendation systems, showing users' 1-5 movie ratings. Empty cells indicate unrated movies. This matrix is key for collaborative filtering, which predicts user preferences based on similar ratings.

Hybrid Filtering

Hybrid filtering combines multiple filtering methods to provide more accurate and relevant recommendations. The main goal of hybrid filtering is to leverage the strengths and mitigate the weaknesses of individual filtering approaches.

Hybrid filtering can integrate these methods by weighting the output of different methods, switching between methods based on context, or using a cascading approach where one method refines the output of another.

2.2.2 LLM-Based Recommendation

The recommendation systems community has made great progress in recent years, but conventional recommendation models (CRM) still face challenges. These include a lack of open-domain knowledge and struggles in fully understanding user preferences and motivations [27]. On the other hand, LLM systems have shown impressive results in a range of NLP tasks due to their extensive knowledge base, logical reasoning capabilities, and understanding of human culture [28, 29]. As a result, the development of LLMs offers exciting new possibilities for improving recommendation systems. There is a promising direction for research in integrating LLMs into recommendation systems to exploit their broad knowledge and capabilities, thereby addressing the shortcomings of conventional recommendation models [30].

Overview of LLMs in Recommendation System

Extensive research has explored the integration of LLMs into recommendation systems, focusing on two key aspects, *where* and *how*, based on the classification by [30], as shown in Fig 2.5.

- The *where* aspect identifies the specific stages within the recommendation system (RS) where an LLM can be integrated. These stages include feature engineering, feature encoding, the scoring or ranking function, item generation, user interaction, and the pipeline controller.
- The *how* aspect examines the methods of adapting LLMs for RS, focusing on two main strategies. The first strategy considers updating the parameters of LLMs during the training phase (injecting collaborative knowledge from a data-centric aspect). The second strategy explores the integration of CRM during the inference phase (injecting collaborative knowledge from a model-centric aspect).

In the following sections, the literature review for both aspects will be discussed in more detail.

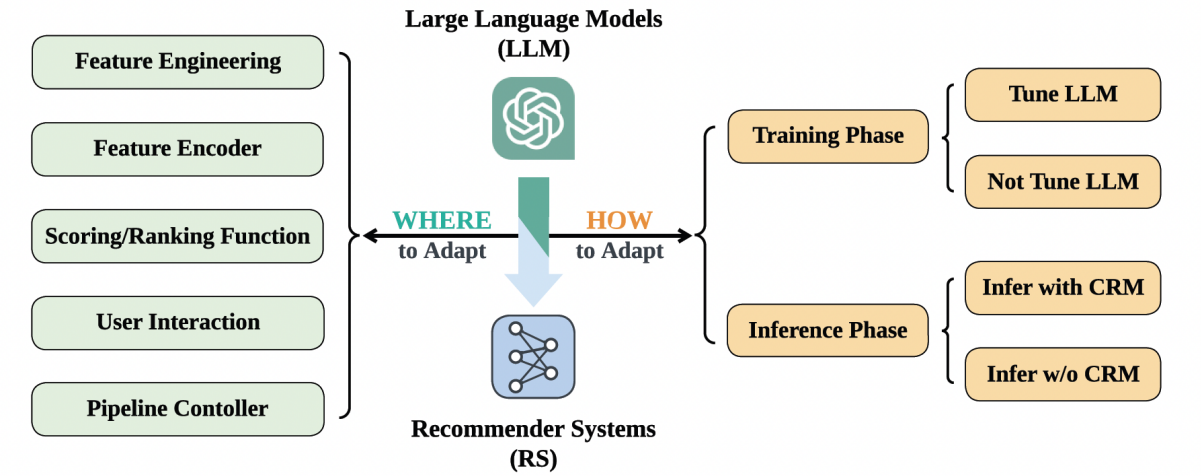


Figure 2.5: Overview of LLMs in RS

Where: Stages of LLMs Integration in Recommendation Systems

LLMs can be adapted at different stages of the recommendation pipeline to improve performance and personalisation. These stages include feature engineering, feature encoding, scoring or ranking functions, user interaction, and controlling the recommendation pipeline [30]. Figure 2.6 shows relevant papers that incorporate LLMs at each of the stages mentioned. We will delve into details on feature engineering and scoring or ranking functions, as feature encoding, user interaction and controlling recommendation pipeline are beyond the scope of this thesis.

Feature engineering with LLMs involves augmenting traditional recommendation systems with rich, contextual features derived from extensive language model knowledge bases. Domain-specific models such as KAR [27] use LLMs to generate insightful features that incorporate user-side preferences and item-specific factual knowledge. CUP [31] uses ChatGPT to distil user ratings into concise keywords, effectively summarising complex user profiles within limited token constraints suitable for further processing by smaller models such as BERT. CUP outperforms state-of-the-art baselines such as DeepCoNN [32] (which uses two coupled neural networks to jointly learn user behaviour and item properties from reviews), P5 [33] (which introduces a foundational model for various downstream recommendation tasks, with all data transformed into natural language sequences and adaptive personalised prompts), and LLM-Rec [34] (which

provides personalised text-based recommendations through prompting), especially in the data-poor setting.

LLMs can directly influence the scoring and ranking stages of recommendation systems by acting as complex function approximators that predict how likely a user is to prefer a specific item. Specifically, at these stages, the goal is to generate a ranked list of items, denoted as $[i_k]_{k=1}^{N}$ (where $i_k \in \mathcal{I}$), for a target user $u \in \mathcal{U}$. $\mathcal{I}$ and $\mathcal{U}$ represent the universal set of items and users respectively. LLMs serve as a pointwise function, denoted by $F(u, i)$, to evaluate each candidate item i for user u. The process involves first constructing a candidate set C of items generated by a pre-filter function C, and then sorting these candidate items based on their scores to obtain the final ranking. Note that we adopt the formulation from [30].

$$
\begin{aligned}
C &\leftarrow \text{Pre-filter}(u, \mathcal{I}), N < |C| \\
[i_k]_{k=1}^{N} &\leftarrow \text{Sort}(\{F(u, i) \mid i \in C\})
\end{aligned}
$$

LLMs process input in the form of discrete textual prompt x, outputting a target token $\hat{t}$ as follows:

$$
\begin{aligned}
h &= \text{LLM}(x), \\
s &= \text{LM\_Head}(h) \in \mathbb{R}^V, \\
p &= \text{Softmax}(s) \in \mathbb{R}^V, \\
\hat{t} &\sim p,
\end{aligned}
$$

where h represents the final embedding, V the vocabulary size, and $\hat{t}$ is the token predicted from the probability distribution p.

To adapt LLMs to output real numbers as scores instead of discrete tokens, three main methods are used. The first approach abandons the traditional language model decoder head, LM_Head(h), and uses a multi-layer perceptron (MLP) to map the final representation of the LLM to a score, i.e. $F(u, i) = MLP(h)$. An example of research using this approach is ClickPrompt [35]. ClickPrompt introduces a novel framework for click-through rate (CTR) prediction, where CTR models serve as soft prompt generators for pre-trained language models (PLMs). The highlight of this model is that it models the mutual interaction and explicit alignment between collaborative and semantic knowledge via the soft prompt interface. In this project, we use a similar approach by adding an MLP to map the final hidden state of a BERT encoder to a score.

The second approach also discards the decoder head and adopts a two-tower structure. By using

two separate towers (i.e. neural networks) for user and item representations, and computing the preference score using distance metrics, d:

$$\hat{y} = F(u, i) = d\left(T_u\left(x_u\right), T_i\left(x_i\right)\right),$$

where $T_u(\cdot)$ and $T_i(\cdot)$ are the user and item towers, using LLM to extract the useful knowledge representations from both the user and item texts (i.e. x_u and x_i). Examples of research using this method are, CUP [31] and RecFormer [36].

The third approach includes methods that either integrate additional prediction modules with the LLM, extract the softmax probabilities of label tokens to compute scores directly [37], or transform the item scoring task into a binary question answering problem and then use bimodal softmax to compute scores from the LLM outputs [38].

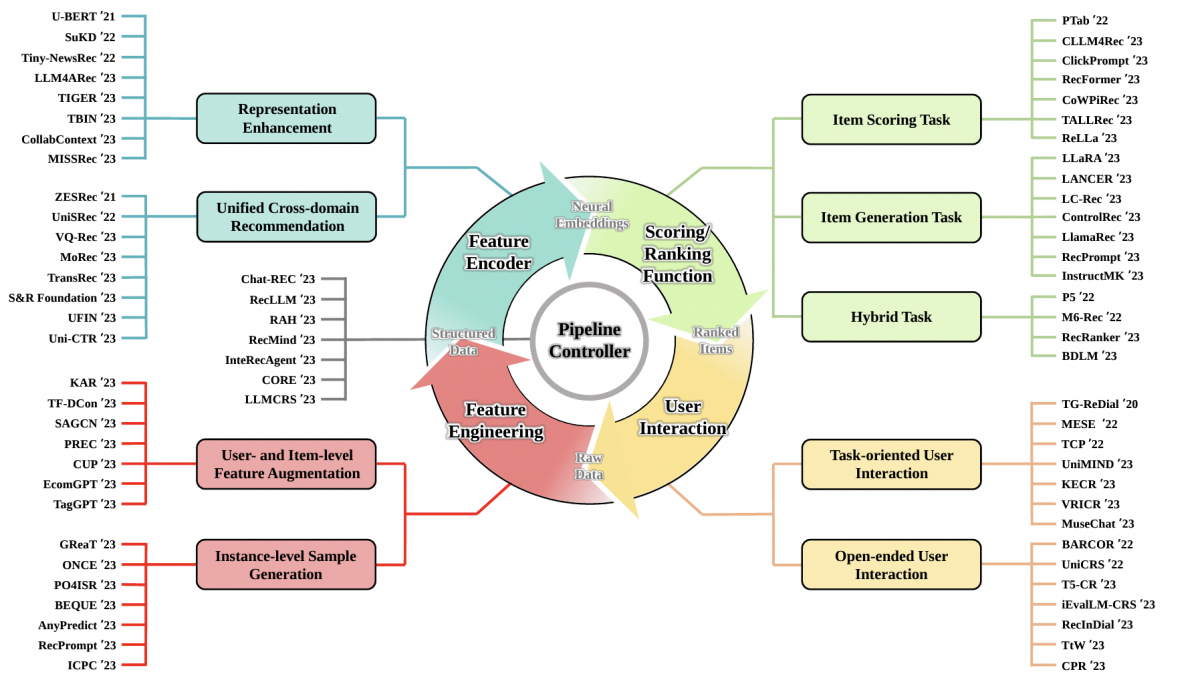


Figure 2.6: Researches of Integrating LLMs into Different Stages of RS pipeline

How: Trends in Adapting LLMs to Recommendation System

Figure 2.7 [30] illustrates the development trends in the integration of LLMs into RS. The figure is divided into four quadrants, each representing a different integration approach. Each circle corresponds to a specific piece of research, where the size of the circle indicates the size of the largest LLM used, and the colour represents the baseline models that it outperformed.

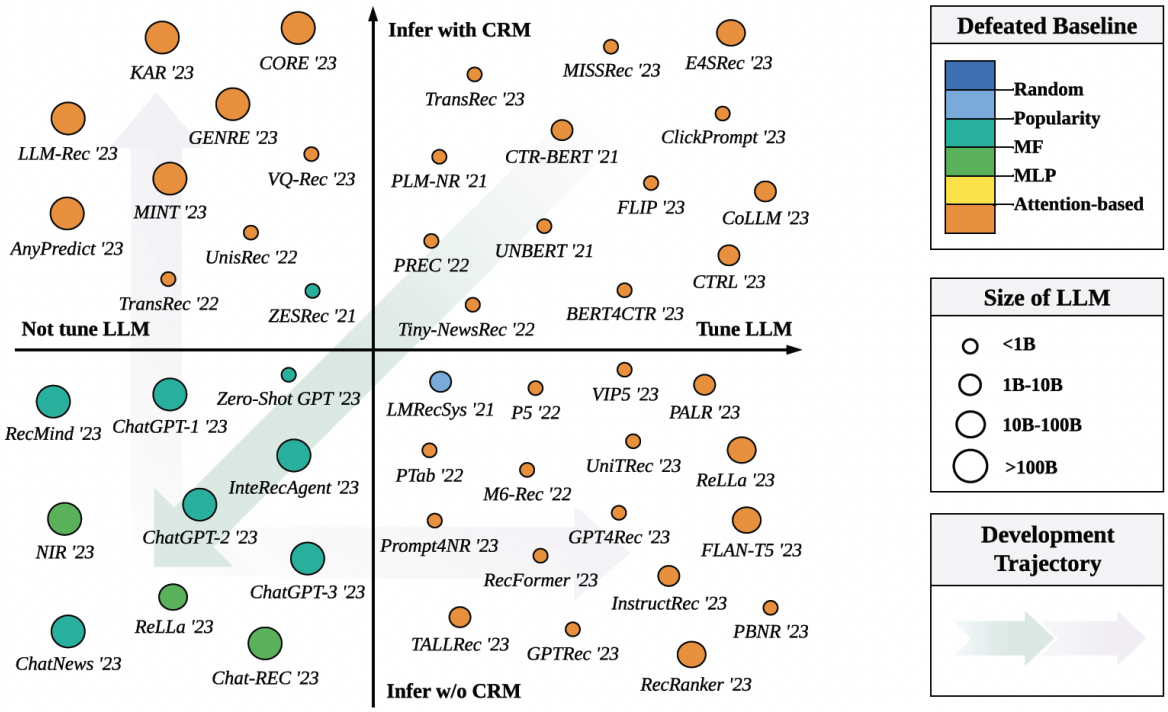


Figure 2.7: Trend of Development of LLM-Based Recommendation System

Works falling within Quadrant 1 not only fine-tune LLMs using domain-specific data, but also incorporate CRM during the inference phrase, and can be classified into two main stages. Initially, relatively smaller pre-trained language models like BERT [2] and GPT2 [39] were mainly used to improve feature encoding for CRMs with additional textual data, improving both semantic and recommendation quality in news recommendation [40], web search [41] and e-commercial advertising [42, 43]. With the rise of LLMs, the trend shifted towards the integration of billion-level LLMs such as ChatGLM [44] and LLama [5], which have been finetuned using parameter-efficient methods such as LoRA [45] and prompt tuning [46] in order to manage computational costs, due to their superior reasoning and instruction-following capabilities.

Research in Quadrant 3 shifts away from traditional recommendation models and instead uses a frozen LLM as the recommendation. Despite using methods such as zero-shot, few-shot learning and careful prompt engineering, early work using the ChatGPT API to perform point-wise, pair-wise and list-wise ranking still has considerable room for improvement [47]. Studies such as ReLLa [48] and RecMind [49] improve this by enabling semantic user behavior retrieval and allowing LLMs to access external knowledge bases like SQL databases and web engines. Despite these advances, the results shown in Fig. 2.7 indicate that the over-generalised knowledge of

LLMs and the lack of in-domain information lead to suboptimal performance with frozen LLMs and inference without CRM (only outperforming MF or MLP baselines). This highlights the need for methods that integrate domain-specific collaborative knowledge from recommendation systems into the LLM training process. This can be achieved either by incorporating CRM into the inference process (quadrant 2) or by tuning the LLM based on domain-specific data (quadrant 4).

In Quadrant 2, works use LLMs without fine-tuning, leveraging their rich semantic and reasoning abilities to enhance CRMs. Unlike Quadrant 1, these models, such as a fixed BERT, provide transferable text embeddings that are integrated into different projection layers to support CRM, as seen in ZESRec [50] and VQ-Rec [51]. As model sizes increase, more sophisticated applications such as SAGCN [52] use LLMs to extract semantic aspect-aware reviews using a chain-based prompting approach and integrate them into graph convolutional networks. When evaluated on three Amazon review datasets [53], SAGCN outperformed state-of-the-art GCN-based recommendation models. Although the LLM is frozen, it plays a crucial supporting role throughout the CRM process, from feature engineering to user interaction [54, 55, 27].

In Quadrant 4, researchers use fine-tuned LLMs without integrating them into CRMs. LLMSeqPrompt [56] introduces domain knowledge into the OpenAI `ada` model by fine-tuning it with prompts that include lists of products from ongoing online shopping sessions, improving recommendation accuracy. As a generative model, LLMSeqPrompt generates hallucinated products, raising the need to ground the generated products in real ones by computing the similarity between them. Recformer [36], a novel bi-directional Transformer model, represents products as key-value attribute pairs and applies a two-stage fine-tuning process with accurate supervision for downstream tasks based on masked language modelling loss. By incorporating domain knowledge through fine-tuning on task-specific data, Recformer demonstrates superior performance on Amazon Review datasets [56], particularly in settings such as zero-shot and cold-start item recommendations, significantly outperforming existing methods. These studies show that domain-specific fine-tuning of LLMs can improve recommendation performance, especially in e-commerce applications, without the need for CRM integration during inference.

2.3 Background on Identifying Substitute Products

Research on identifying product relationships in recommendation systems is well developed, with many studies considering both complementary and substitute products simultaneously due to their close interrelationship [57]. Although this project focuses specifically on substitution, the findings of studies that consider both types of relationships are also discussed. These studies provide valuable alternative approaches to identifying substitutability. In this section we provide a brief overview of the key findings from the related works.

Early Stage

The initial effort to identify substitutes and complements on e-commerce websites involved a tagging algorithm described in [58]. This algorithm identifies stages in the purchase lifecycle and calculates scores based on users' purchase and browsing data to determine relationships between products. However, a significant limitation of this algorithm is its reliance on manual tagging, which is impractical for large datasets.

Graph-based Approach

Graphs can effectively represent relationships between products, as the connectivity between two nodes can be transformed into the proximity between their learned embeddings of products, as shown in [59, 60]. This approach allows co-purchased products to be represented as linked nodes in a product co-purchase graph. Sceptre [61], a model that uses topic modelling, Latent Dirichlet Allocation (LDA) [62], to analyse texts from product reviews and cluster them into topics representing product features. Each product is represented by a topic vector, which is used in a link prediction model to determine relationships between products, such as whether they are substitutes or complements. The model not only predicts these relationships but also discovers micro-categories of closely related products by leveraging a hierarchical structure of product categories. The relationships are visualised as a product graph, which is trained on data such as co-purchasing patterns from Amazon to predict and connect related products in a large network.

Building on the datasets constructed by Sceptre, another model, SPEM [57], incorporates the concepts of *two substitutable products tend to have similar complementary products* and *two complementary products must not have a substitutable relationship*. These concepts are captured in a product co-purchase graph as second-order proximity and negative first-order proximity, respectively. SPEM preserves second-order proximity by using an unsupervised deep autoencoder, while applying a supervised pairwise constraint on negative first-order proximity. In addition,

the model employs tree-based semantic similarity modelling using category tags within the hierarchy. SPEM was evaluated on seven real-life datasets from Amazon and results show the superiority of it in inferring substitutable products against state-of-the-art baselines.

Probabilistic Approach

Fewer studies have used a probabilistic approach to this problem than a graph-based approach. One such example is the Bayesian model SHOPPER, designed to analyse consumer behaviour in sequential shopping decisions [63]. This model is also able to capture the complex interactions between products and how certain items can act as substitutes or complements to others in a consumer's shopping basket. The model works by treating each purchase decision as a choice from a set of available items, with latent variables representing the attributes of those items. It takes into account various factors such as user preferences, seasonal effects, price and item popularity. A key innovation of the SHOPPER model is its ability to 'think ahead', not only considering the current choice, but also anticipating subsequent decisions within the same shopping trip.

2.4 Chapter Summary

This chapter reviews the evolution of LLMs, with a particular focus on how advances have improved their performance. It also introduces the core techniques of recommendation systems, namely content-based, collaborative and hybrid filtering. Furthermore, it discusses recent work on integrating LLMs into the feature engineering and scoring stages of the recommendation process by explaining the mathematical formulations and methods for adapting LLM token outputs into scores. The chapter also introduces four quadrants that classify the integration methods based on whether LLMs are fine-tuned and whether inference is performed with or without CRM. Finally, the chapter examines historical methods used to identify substitutes, such as graph-based and probabilistic approaches. Collectively, these sections provide a solid foundation for understanding the methods used later in this project.

Chapter 3

Datasets

This chapter provides a comprehensive overview of the datasets used in this project. It begins with the details of the datasets, including descriptions of the data sources, the pre-processing methods used to ensure data quality, and the analytical techniques used to examine the data. It also presents an analysis of the datasets, focusing on aspects such as product division, the distribution of substitute counts, and the Bayesian acceptance rate distribution. These sections collectively establish the foundational data framework and provide insights that are essential for developing a tailored recommendation system that effectively adapts to the characteristics of the data.

3.1 Datasets

3.1.1 Datasets Description

This project uses four datasets sampled over a period of 9 weeks, from week 38 to week 46:

1. **Tesco-Product**: This dataset contains comprehensive information on all products. The dataset provided for this project contains 38 columns, not all of which are used. The columns used in this project are shown in the table. 3.1.

2. **Tesco-Online**: This dataset is collected from users' historical online interactions. If a product is not available online, another product is recommended alongside the unavailable one. A mock example is shown in Table 3.2, where the number of impressions refers to the number of times the recommended substitute is displayed to the user, and the number of adds refers to the number of times the recommended substitute is added to the basket.

3. **Tesco-Offline**: This dataset contains users' offline interactions. If a product ordered online is unavailable, a colleague in the store will select a substitute for the unavailable product. A mock example is shown in table 3.3, where the number of picked represents the number of times this product was chosen as a substitute, while the number of accepted represents the number of times consumers accepted this substitute product on delivery.

4. **Tesco-Candidate**: This dataset is created by Tesco's data science team. It first includes all substitution pairs with historical substitution data, and then filters them by various factors, such as the availability of substitute products in offline stores on the day. A mock example is shown in Table 3.4. Tesco-Candidate is used because candidate generation is beyond the scope of this project. Our focus is on improving performance given the candidate pool.

Table 3.1: Tesco-Product Dataset Fields

Column Name	**Description**
tpnb	Tesco Product Number
live_product	Indicator if the product is available in the week
commercial_hierarchy_division_code	Code representing the commercial hierarchy division
description	Product title
item_weight	Weight of the product
weight_unit	Unit of the product weight
marketing_text	Marketing text associated with the product
average_price	Average price of the product, used because the price fluctuates in a week
scraped_description	Extra description scraped from external sources

Table 3.2: Tesco-Online Dataset: Mock Data

original_tpnb	**sub_tpnb**	**number_of_impressions**	**number_of_adds**
123456	234567	1000	4
234567	345678	300	2
345678	456789	2000	5

Table 3.3: Tesco-Offline Dataset: Mock Data

original_tpnb	**sub_tpnb**	**number_of_picked**	**number_of_accepted**
123456	234567	50	45
234567	345678	30	25
345678	456789	20	19

Table 3.4: Tesco-Candidate Dataset: Mock Data

original_tpnb	**sub_tpnb**	**category**
123456	234567	Food
234567	345678	Alcohol
345678	456789	Drinks

3.1.2 Datasets Pre-processing

We performed two types of preprocessing on the dataset: filtering and filling.

We first filtered the Tesco-Product dataset to remove products that were not live at any point during the 9-week period, significantly reducing the time needed to retrieve additional descriptions and for LLMs to process the data (from 65k to 36k products). Next, we filtered out substitution pairs that had the same original and substituted tpnbs in both Tesco-Online and Tesco-Offline datasets. This duplication occurs because the same item can appear on different pages of the website, such as mango chunks and mango chunks meal deal.

We observed that Tesco-Offline dataset has low product coverage. To be more specific, in a typical test week, around 30% of products have no historical substitution pair information. To address this, we fill the missing data with information from Tesco-Online, reducing the missing data to approximately 2%. This process results in the Tesco-Hybrid dataset.

However, there is a notable difference between online and offline data. The ratio of 'adds to impressions' online is much lower than the ratio of 'accepted to picked' offline. There are two reasons for this difference:

- User Base Size: Online users are a larger population, representing a more diverse set of consumer tastes and preferences, making it more difficult to recommend suitable substitutes.

- Impressions vs. Picks: Impressions indicate how many consumers have viewed the product online, whereas picks represent the actual delivery of the product to the customer's door. Although many consumers may view a substitute, the ease of exploring other alternatives online often leads to fewer items being added to their shopping baskets.

We noticed that around 30 % products are missing product information, making accurate recommendations challenging for the model. To address this issue, we will implement a Retrieval Augmented Generation (RAG) module in our pipeline. This module will enhance the model's ability to make accurate recommendations by retrieving relevant information from external sources. Further details of this implementation are explained in Chapter 4.

3.1.3 Datasets Analysis

Datasets Summary

1. **Tesco-Product**: This dataset contains all the required product information, such as product title, price and category. After pre-processing it contains a total of 36.5k live products.

2. **Tesco-Offline**: This dataset is used for training and evaluation, this dataset contains 243k unique original and substitute pairs after preprocessing. It contains important information about the acceptance rate of the substitutes.

3. **Tesco-Online**: This dataset is used for training, and contains 5.6 million unique original and substitute pairs after pre-processing, with broader product coverage, ideal for filling in missing records in Tesco-Offline.

4. **Tesco-Candidate**: This dataset is used during evaluation and contains a total of 759k original and substitute pairs collected over 9 weeks. The models first rank these candidate pairs before evaluating them against Tesco-Offline. This step helps to assess the effectiveness of the models in a simulated real world scenario.

5. **Tesco-Hybrid**: This dataset is generated in this project, by filling in missing products in Tesco-Offline from Tesco-Online. It consists of 776k unique original and substitute pairs and improves the coverage of the Tesco-Offline dataset.

The following sections will analyze three aspects of the datasets: product division distribution, substitute count distribution, and acceptance rate distribution.

Product Division

Table 3.5 illustrates the distribution of products across different divisions in the Tesco-Product dataset. A detailed understanding of these divisions is important for the LLM's prompting stage and for the fine-tuning of the cross-encoder.

We design prompts tailored to each division since product characteristics differ between divisions. For example, grape variety and country of origin are important in the wine category, while these factors are less relevant in categories such as snacks. Few-shot learning can be used to provide the LLM with division-specific examples, helping it recognize key features unique to each division. This approach ensures more structured and relevant summaries of division-specific features.

Training the cross-encoder with division-specific data is essential to capture the nuanced details and preferences unique to each product division. This targeted fine-tuning approach leads to more accurate and contextually appropriate product substitutions. For example, in the beverage category, understanding flavour profiles is critical, while in cleaning products, efficacy against specific types of stains or surfaces may be more relevant. By tailoring the training process to these details, the cross-encoder learns to effectively prioritise the relevant features, resulting in better-targeted and more satisfactory substitution recommendations.

Since our method is division-specific, we will use the distribution of divisions to select those that will be included in this project. As shown in Table 3.5, Division Z accounts for a very small percentage (with only one product), so it will be excluded. Due to limited computational resources, we will select one division from the larger divisions of similar size. Specifically, we will select five divisions in total: N in G and N, F in F and H, and W, B, and T. The reason for not selecting Division V will be explained in Section 3.1.3.

Table 3.5: Division Statistics

Division Name	Division Code	Percentage (%)
AMBIENT DRY GROCERY	G	27.531
NON FOODS GROCERY	N	22.617
FRESH FOODS	F	21.425
HOME & WEAR	H	15.823
WINES & SPIRITS	W	6.152
BREAD/BAKERY & REST.	B	4.419
TOBACCO KIOSKS	T	1.558
TESCO MOBILE	V	0.473
MISCELLANEOUS ITEMS	Z	0.003

Substitute Count Distribution

Figure 3.1 shows the distribution of substitute counts for each product across Tesco-Offline, Tesco-Candidate and Tesco-Online datasets. The frequency of these substitute counts generally decreases exponentially as the number of substitutes increases. Specifically, there are over 10,000 products with fewer than 20, 100, and 1000 substitutes in the Tesco-Offline, Tesco-Candidate, and Tesco-Online datasets, respectively.

Understanding the distribution of substitute counts offers valuable insights:

- **Tesco-Offline**: This distribution of substitute counts collected from offline helps guide the recommendation system to offer a reasonable number of alternatives. For products with many substitutes recorded historically, the system may present more options, while for those with fewer substitutes, it may be more conservative. Forcing the system to generate a list of substitutions that is significantly larger than the number of substitutions that occurred in the historical data could lead to irrelevant recommendations, reducing model performance. This insight helps in selecting the appropriate k value for evaluation metrics ncdg. As the average number of substitutions is around 9.6 (shown in the title of the first plot in Figure 3.1), we will use NDCG@10 for this project.
- **Tesco-Offline vs. Tesco-Candidate**: The distribution of substitute count in Tesco-Candidate can validate the correctness of this dataset as a candidate pool. As a set of potential candidates for substitution, it should cover a slightly broader range than the real historical dataset, Tesco-Offline. The histograms confirm this, with mean and max substitute counts being 9.6 and 146 for Tesco-Offline, and 15.33 and 505 for Tesco-Candidate, respectively.
- **Tesco-Offline vs. Tesco-Online**: Comparing these histograms shows that the distribution of substitute counts in Tesco-Online is more varied than in Tesco-Offline, with some products having even over 4000 possible substitutions. This suggests that there is a wider range of product-substitution pairs available online. This finding leads to the pre-processing strategy of using Tesco Online to fill in missing products in Tesco Offline.

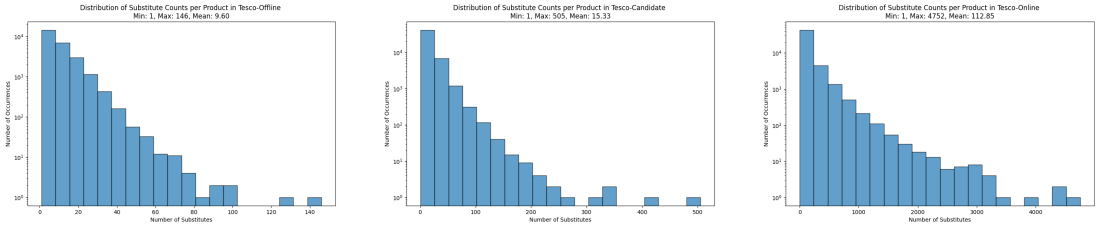


Figure 3.1: Distribution of Substitute Counts for Online, Offline, and Candidate Data

Bayesian Acceptance Rate Distribution

The traditional acceptance rate is calculated as the ratio of the number of accepted substitutions to the number of picked substitutions:

$$\text{Traditional Acceptance Rate} = \frac{\text{Number of Accepted Substitutions}}{\text{Number of Picked Substitutions}}$$

However, this formula fails to consider the total number of observations, which can result in inaccurate conclusions. For example, if a substitute is selected only once and accepted, it would show a 100% acceptance rate. However, this is misleading when compared to a substitute picked 100 times and accepted 99 times, which shows a 99% acceptance rate. Despite the lower percentage, the latter is a more reliable measure due to the larger sample size.

To solve this problem, we introduce the Bayesian acceptance rate. This approach starts by estimating a prior from historical user data, then updates it with observed data to get a posterior distribution, and the Bayesian score is the mean of this posterior.

The substitution problem can be modeled with a binomial distribution, where successes are acceptances and failures are rejections. Since the conjugate prior for a binomial distribution is the beta distribution, our goal is to use a beta prior that reflects the relative success rate of accepted picks while remaining non-informative.

Let k_{sum} be the total number of accepted picks in a week, and n_{sum} be the total number of picks. The beta prior is constructed as follows to reflect the relative success rate of accepted picks:

$$\beta(a, b) = \begin{cases} \beta\left(\frac{n_{\text{sum}}}{k_{\text{sum}}}, 1\right) & \text{if } \frac{k_{\text{sum}}}{n_{\text{sum}}} > 0.5 \\ \beta\left(1, \frac{n_{\text{sum}}}{n_{\text{sum}} - k_{\text{sum}}}\right) & \text{otherwise} \end{cases} \tag{3.1}$$

The Bayesian acceptance rate is then calculated as the mean of the beta distribution:

$$\text{Bayesian Acceptance Rate} = \frac{a}{a + b}$$

For example, if there are a total of 200 accepted picks out of 1000 in a week, we should use a Beta distribution, $\text{Beta}(1, 5)$, as our prior which reflects the ratio of accepted picks. For each product, we then update this Bayesian prior with the observed number of accepted picks (k) and rejected picks ($n - k$), resulting in an updated Beta distribution, $\text{Beta}(1 + k, 5 + n - k)$. This method allows us to refine our acceptance rate estimates dynamically based on observed data. As shown in Fig 3.2, the Bayesian score for accepting 99 out of 100 picks is much higher than for accepting 1 out of 1, which successfully address the problem.

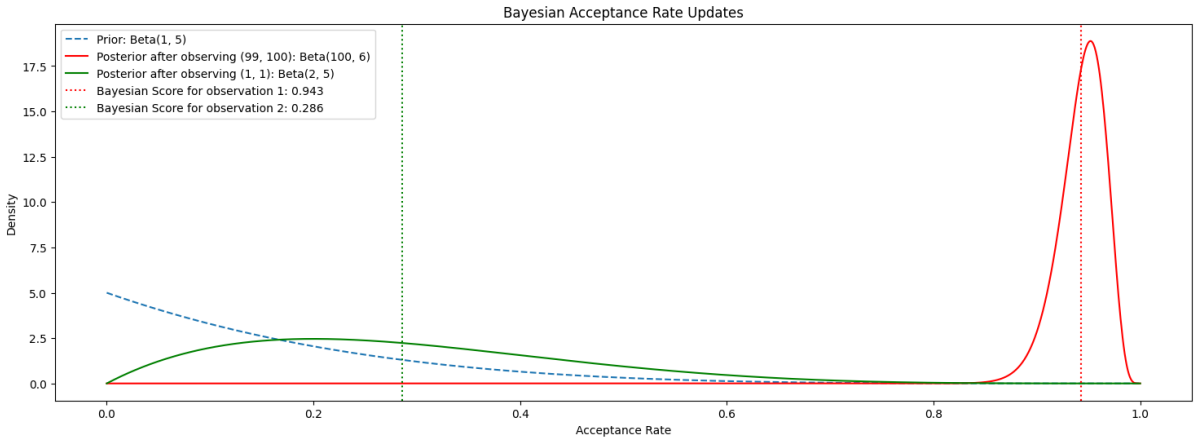


Figure 3.2: Bayesian Acceptance Rate Updates

Figure 3.3 compares the Bayesian score distribution in Tesco-Hybrid across different divisions. Analyzing this can provide us with important insights. Although this project only focuses on five divisions, plotting the score distribution for all divisions could benefit future work at Tesco and provide a more comprehensive understanding of the overall score distribution.

First, understanding the distribution of scores helps to identify biases in the dataset. For example, we can see that most product-substitution pairs are labelled with scores around 0.1 to 0.2 or around 0.5; the dataset is biased towards lower scores. This knowledge is crucial for preparing our fine-tuning dataset for the cross-encoder.

Second, analysing the distribution of scores helps to devise strategies to balance the dataset, leading to more robust learning. As more scores are below 0.5, using negative sampling can help to produce a more balanced dataset. We could also increase the coverage of the dataset or use data augmentation methods. However, these strategies are beyond the scope of this project and would be undertaken if additional computational resources and time were available.

Finally, examining the score distribution can reveal data quality issues. Large clusters of scores at certain intervals or unexpected gaps might indicate problems with the dataset's suitability for fine-tuning. Specifically, the score data for Division V are very imbalanced and sparse, making them unsuitable for training the cross-encoder. Consequently, we will exclude them from the training dataset.

By thoroughly analysing the score distributions, we can make informed decisions on dataset preparation, sampling strategies and data quality, and eventually improve the performance of our model.

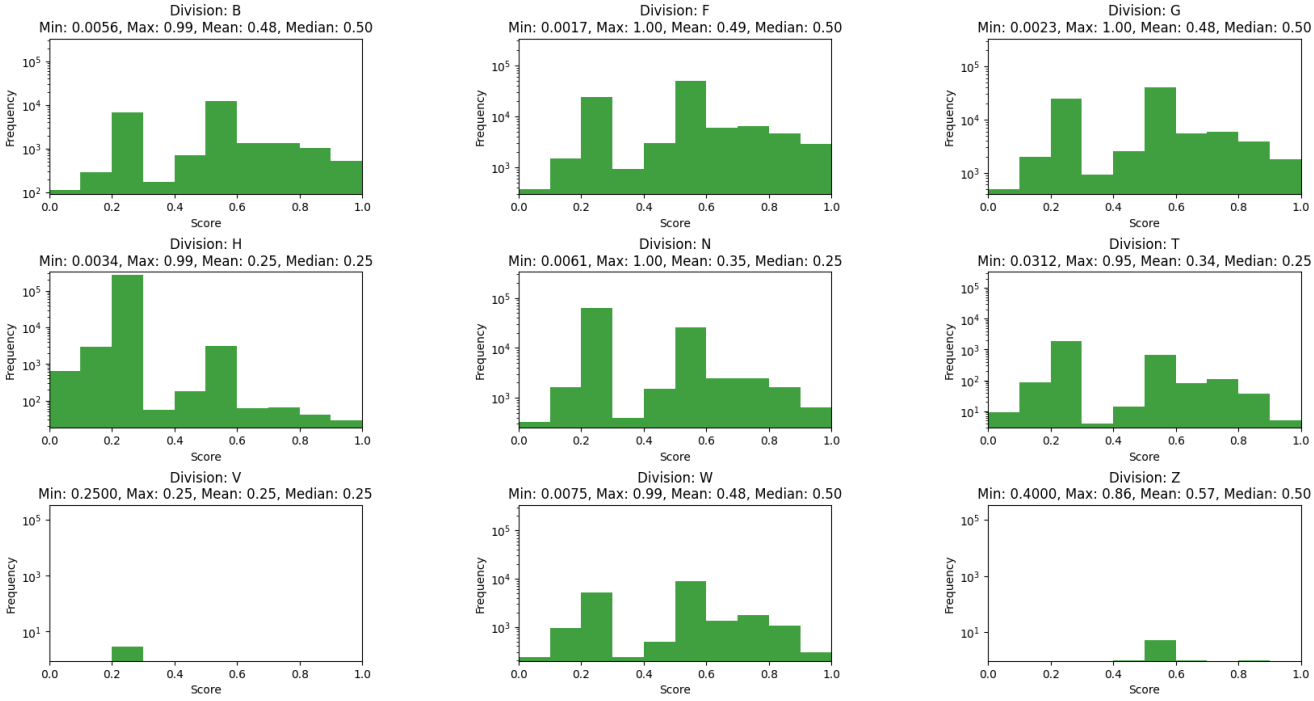


Figure 3.3: Bayesian Acceptance Rate Distribution in Tesco-Hybrid Across Different Divisions

3.2 Chapter Summary

This chapter provides a detailed examination of the datasets used and presents mock examples for the datasets to improve understanding of their structure. We improve the quality of these datasets by filtering out non-live and incorrect products. In addition, we enrich the offline dataset by filling in missing information from the online dataset. We analyse the datasets from three different perspectives: product division distribution, substitution count distribution and Bayesian acceptance rate distribution. The insights derived from these analyses are used to design a method that is tailored to the characteristics of the datasets.

Chapter 4

Methodology Design

This chapter presents the design of a recommendation system that leverages the latest advances in NLP to enhance product substitution recommendations. Our methodology involves a dual-component system, consisting of an LLM, `gpt-4o-mini`*, for feature augmentation and a cross-encoder,* `ms-marco-MiniLM-L-6-v2`*, for candidate re-ranking. We begin by discussing the rationale behind the selection of these models, followed by a detailed technical explanation of each component in the pipeline. In particular, we focus on the components of the cross-encoder, detailing how they contribute to an improved comparison between product-substitution pairs.*

Our pipeline consists of two components that work together sequentially to enhance overall performance, as shown in Figure 4.1. The first component, an LLM with RAG (Retrieval-Augmented Generation), acts as a feature augmenter and addresses any missing product information. The second component is a cross-encoder that functions as a candidate re-ranker, generating a ranked list of recommended substitutes by comparing product-substitution pairs from the Tesco-Candidate dataset.

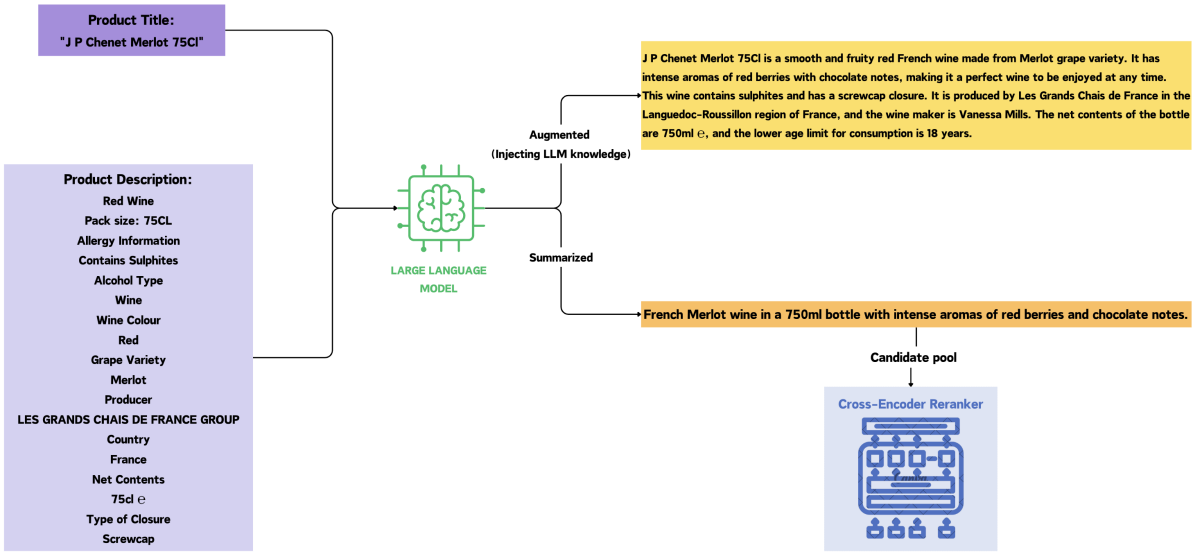


Figure 4.1: Overview of Pipeline

4.1 LLM with RAG as Feature Augmenter

The motivation to implement this component stems from two key observations. First, during exploratory data analysis, we discovered around 30% of products with missing information. Second, we were inspired by the research [27], which emphasises that recommendation system should be designed as an open-world system, proactively acquiring knowledge from the external environment. In addition to retrieving information from external sources, we should leverage the open-world knowledge of LLMs, as they are trained on vast datasets from diverse origins and can contribute their learned knowledge to enhance our model's product understanding.

RAG, or Retrieval-Augmented Generation, is an approach used in NLP that combines two techniques: retrieval and generation. It provides an ideal solution for dealing with missing product information in datasets and providing the recommendation system with updated knowledge from the open world. By retrieving reliable external product information, RAG provides LLMs with additional context, ensuring the generation of more specific and accurate product descriptions while reducing hallucinations. Additionally, by carefully designing prompts, we encourage LLMs to incorporate their internal knowledge of the products into the augmentation process. In the following sections, we will introduce the two stages of RAG in detail.

Retriever

We use a web scraper as our retriever to scrape product information from search engines such as Google and DuckDuckGo. To ensure compliance with data use policies, the scraper has been restricted to data from the official Tesco website only. The scraper is flexible in its capabilities, supporting both server-side scraping using the `requests` and `BeautifulSoup` libraries, and browser-based scraping using `Selenium`. The web scarper we implemented allows for accurate and up-to-date extraction of product details, while adhering to legal data handling standards.

Augmentation with Generative Model

We choose the `gpt-4o-mini` as our generative model, considering the balance between price and accuracy. According to a report by OpenAI, `gpt-4o-mini` outperforms `GPT-3.5 Turbo` and other small models on academic benchmarks in both textual intelligence and multimodal reasoning. In particular, `gpt-4o-mini` scores 82.0% on the MMLU [64], a benchmark of textual intelligence and reasoning. Comparing to its counterparts, the `Gemini Flash` [65] scored 77.9% and the Claude Haiku [66] scored 73.8%.

Our project requires the generative model to have strong textual intelligence and reasoning

capabilities to understand the main functionalities and key features of the products. In addition, the model must have excellent interpretation and summarisation capabilities. `gpt-4o-mini` meets these requirements and supports text and vision in its API, allowing text, image and video inputs, opening up the possibility of using product images in substitution decisions. With knowledge updated until October 2023, it ensures that information is up to date.

In terms of cost, `gpt-4o-mini` is offered at 15 cents per million input tokens and 60 cents per million output tokens, making it significantly more affordable than previous frontier models and over 60% cheaper than `GPT-3.5 Turbo`. Its low cost and latency enable a wide range of tasks such as chaining, parallelizing multiple API calls and processing large amounts of context. If implemented, this method would be applied to millions of products in Tesco, so we need a cost-efficient model.

4.2 Cross-Encoder as Candidate Re-ranker

The motivation to implement a cross-encoder to rank candidate substitutes instead of prompting LLMs, like GPT-4 and Llama 3, stems from several considerations:

1. Given the size of Tesco's dataset, which includes over 700,000 product-substitute pairs that need to be ranked weekly, using LLMs via an API is not feasible due to daily request limits. Running LLMs on a local machine is also inefficient. For example, using Llama 3 to process this volume of data could take over 20 hours per week on an Apple M3 GPU.

2. Prompting LLMs to rank products often results in unstable rankings, this variability can be mitigated by implementing a two-stage pipeline involving using a probing detection dataset and a Bayesian strategy [67]. However, this approach increases the complexity of the method and is not applicable to this project, given the large size of the dataset.

3. Research [68] has shown that cross-encoders produce competitive results when re-ranking outputs from SPLADE, a first-stage ranker, on the Microsoft MAchine Reading Comprehension (MS MARCO) dataset. This highlights the potential of using cross-encoders as re-rankers to achieve a balance between effectiveness and efficiency. We will investigate further into the actual performance of this approach in later sections.

4. While LLMs require extensive fine-tuning for specific ranking tasks, cross-encoders are generally smaller and easier to fine-tune for similar tasks. This ease of adaptation can provide a quicker setup and potentially lower running costs, making them a preferable option for weekly ranking needs.

In this thesis, we use the cross-encoder `ms-marco-MiniLM-L-6-v2` from Hugging Face, which uses `BertForSequenceClassification` architecture, a variant of BERT with an added classification head. The `ms-marco-MiniLM-L-6-v2` model is pre-trained on the MS MARCO Passage Ranking task, making it well-suited for information retrieval. It takes a query and a list of passages as input, then ranks these passages in decreasing order of relevance. This approach easily adapts to our project: given the description of an original product, we want to generate a ranked list of substitutes based on the similarity of their descriptions to the original. Figure 4.2 provides an overview of how the cross-encoder works.

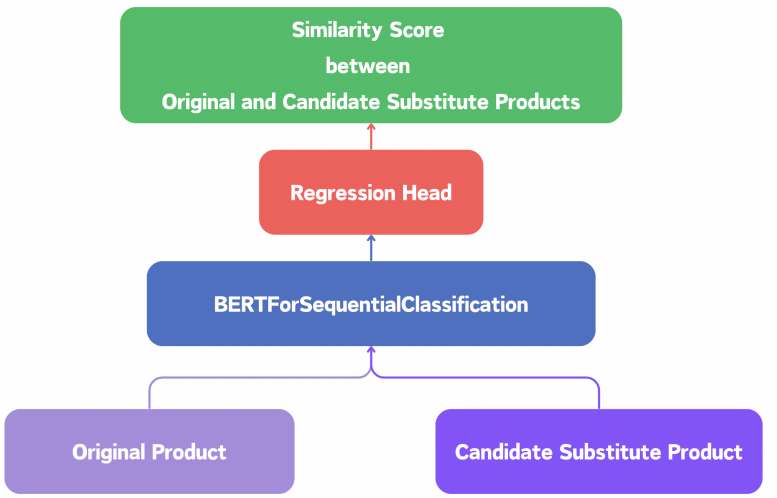


Figure 4.2: Overview of Cross-Encoder

4.2.1 Cross-Encoder Architecture

In this section we will examine each component of the `ms-marco-MiniLM-L-6-v2` model in detail.

Tokenization

Tokenization is the first component of `ms-marco-MiniLM-L-6-v2`. The model uses the BERT tokenizer, which is responsible for breaking down the input text into smaller units. As shown in Figure 4.3, five types of special tokens are used:

- The `[CLS]` token, stands for classification, is used in the next sentence prediction (NSP) task, which determines whether a second sentence logically follows the first sentence. During pre-training, the input to the model consists of a pair of sentences, with the `[CLS]`

token at the beginning of each sentence. Through this process, the `[CLS]` token learns to capture the sentence-level representation.

- The `[MASK]` token is used in masked language modeling (MLM) tasks. During pre-training, tokens in the input sentence is randomly replaced with `[MASK]` and the task is to predict the original tokens based on the context provided by other words. This enables the language models to learn relationship of tokens within a sentence.
- The `[PAD]` token is used to make all input sequences in a batch the same length, as NLP models often require fixed-length inputs.
- The `[SEP]` token indicates the boundary between input sentences, when two sentences are concatenated into a single input sequence.
- The `[UNK]` token represents words or tokens that are not found in the model's vocabulary.

Figure 4.3 illustrates the three types of IDs used:

- **Token Type IDs:** These IDs are used in NSP task to distinguish between tokens belonging to the first or second sentence. In `BERTForSequenceClassification` architecture, all tokens in a sequence are treated equally. Therefore, each token is encoded with a zero.
- **Attention Mask:** By adding padding tokens which do not contribute to the model's computations, all input sentences have equal length. The model ignores these padding tokens by assigning them a zero in the attention mask, while actual content tokens are marked with a one.
- **Input IDs:** These represent the unique identifier for each token within a predefined vocabulary of 30,522 tokens.

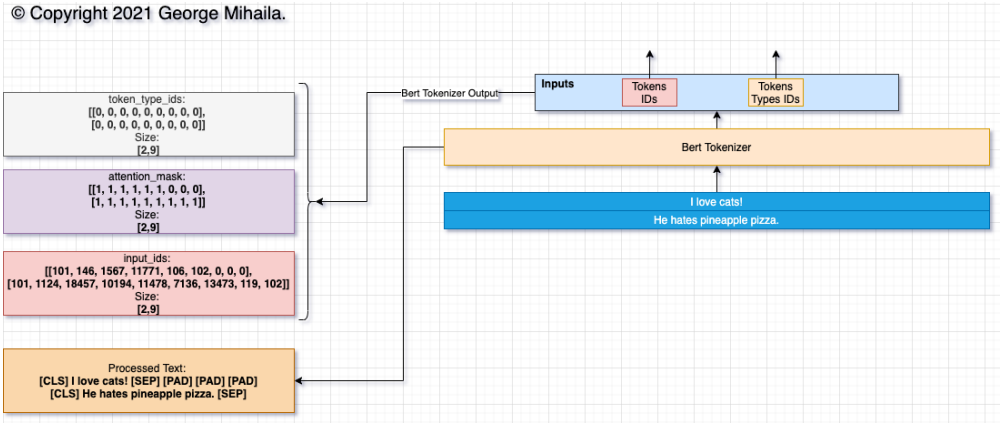


Figure 4.3: Tokenizer

Embedding

Embeddings are numerical vector representations of tokens that allow models to process text through mathematical operations. During training, the model learns to map similar tokens to nearby points in the embedding space, while positioning dissimilar tokens further apart. Once trained, a word embedding lookup table is created, mapping each token to its corresponding embedding. In this case, each token is represented by an embedding vector of 768 dimensions.

As shown in Fig. 4.4, after tokenization by the BERT tokenizer, the token IDs are processed by the embedding layers to generate comprehensive embeddings. The model uses three types of embeddings to capture different aspects of the token information:

- **Word Embeddings**: These embeddings capture the lexical meaning of each token independently of other tokens. To be more specific, each input ID token is replaced with the corresponding vector from the word embedding lookup table (with in total 30522 rows corresponding to the BERT vocabulary size).
- **Token Type Embeddings**: Token type embeddings are generated from token type IDs and help to distinguish between different sentences in the NSP task. However, in the cross-encoder used, the token type IDs are all set to zero, so they have no effect.
- **Positional Embeddings**: Positional embeddings are used to provide the model with information about the position of each token in the sequence. This helps the model to understand the order of the tokens, which is crucial for correctly interpreting phrases and sentences. Here we use the sine and cosine function of different frequencies as the original paper of the transformer [6]:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right)$$

  where $d_{\text{model}} = 768$, pos represents the position of the token within the sequence and i is the index within the positional encoding vector.

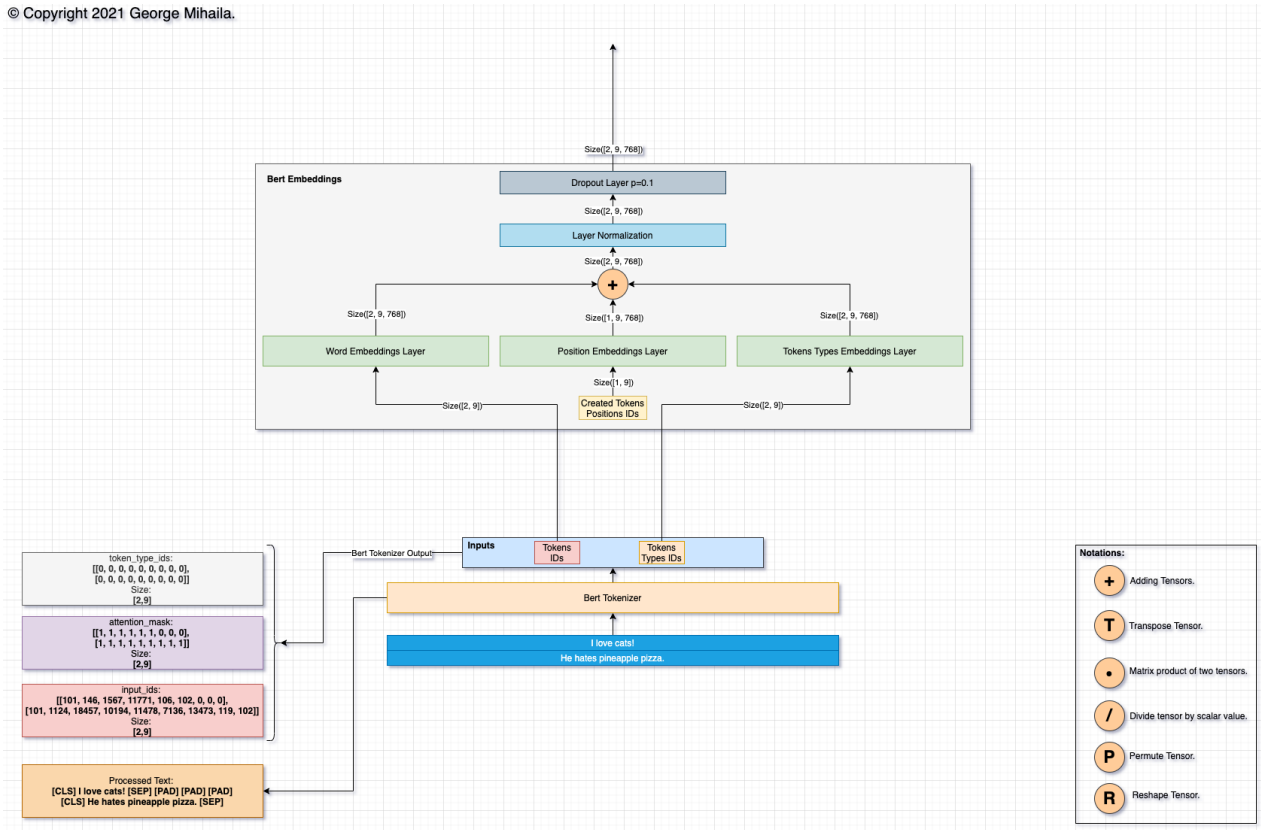


Figure 4.4: Embedding Module

Encoder

The encoder of the `ms-marco-MiniLM-L-6-v2` model consists of 12 transformer layers, each contains two key sub-layers: a multi-headed self-attention module and a position-wise feed-forward network, as shown in Fig 4.5.

For better explanation, we will describe the components of the encoder module in the order that the input sequence is processed. For this project, we are using dimension of embedding $d_{\text{model}} = 768$, number of heads $h = 12$, and dimensions of key and value vectors $d_k = d_v = \frac{d_{\text{model}}}{h} = 64$.

As the sequence passes through each transformer layer, it first processed by the multi-headed self-attention mechanism (illustrated in the BERT Attention block in Fig 4.5). In this process, different parts of the sequence are weighted according to their relevance to other parts. This mechanism was introduced in the original paper by Vaswani et al. (2017) [6].

To be more specific, self-attention scores are calculated using the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

which can be interpreted as a weighted sum of the value vectors based on the similarity between the query and key vectors.

Multi-head attention mechanism enhances model capabilities by parallelizing the attention mechanism across multiple "heads". Each head independently processes the input data using self-attention mechanism, allowing model to focus on different relationships in the sequence. These independent outputs are then concatenated and undergo a final linear transformation. Multi-head attention mechanism leads to richer and more robust representations. Detail computation is shown below:

$$\begin{aligned}\text{MultiHead}(Q, K, V) = &\ \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^\rho \\ \text{where head}_\text{i} = &\ \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)\end{aligned}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $W^\rho \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ is a projection matrix that reshapes the concatenated output back to the original embedding dimension d_{model} . In the context of self-attention, Q, K, and V all originate from the same source, which is X — the matrix containing the embedding vectors of our input sequence.

The output of from multi-head self-attention module is then normalized and passed to the position-wise feed-forward network (refer to Bert Self Output block in Fig 4.5). The position-wise feed-forward network is the second main component of a transformer layer. It first project the input to the higher embedding size (768 to 3072), then pass through a Gaussian Error Linear Units (GeLU) activation function and project back down to original embedding size. Expanding to a higher dimensional space allows the model to capture more nuanced details in the data. Hendrycks et al. (2016) [69] demonstrated that GeLU activation functions enhance performance in NLP tasks compared to Rectified Linear Unit (ReLU) and Exponential Linear Unit (ELU) activations, potentially leading to improved model performance . By applying dropout, a skip connection, and performing layer normalization in the following steps, helps to stabilize the learning process and mitigates the vanishing gradient problem during training.

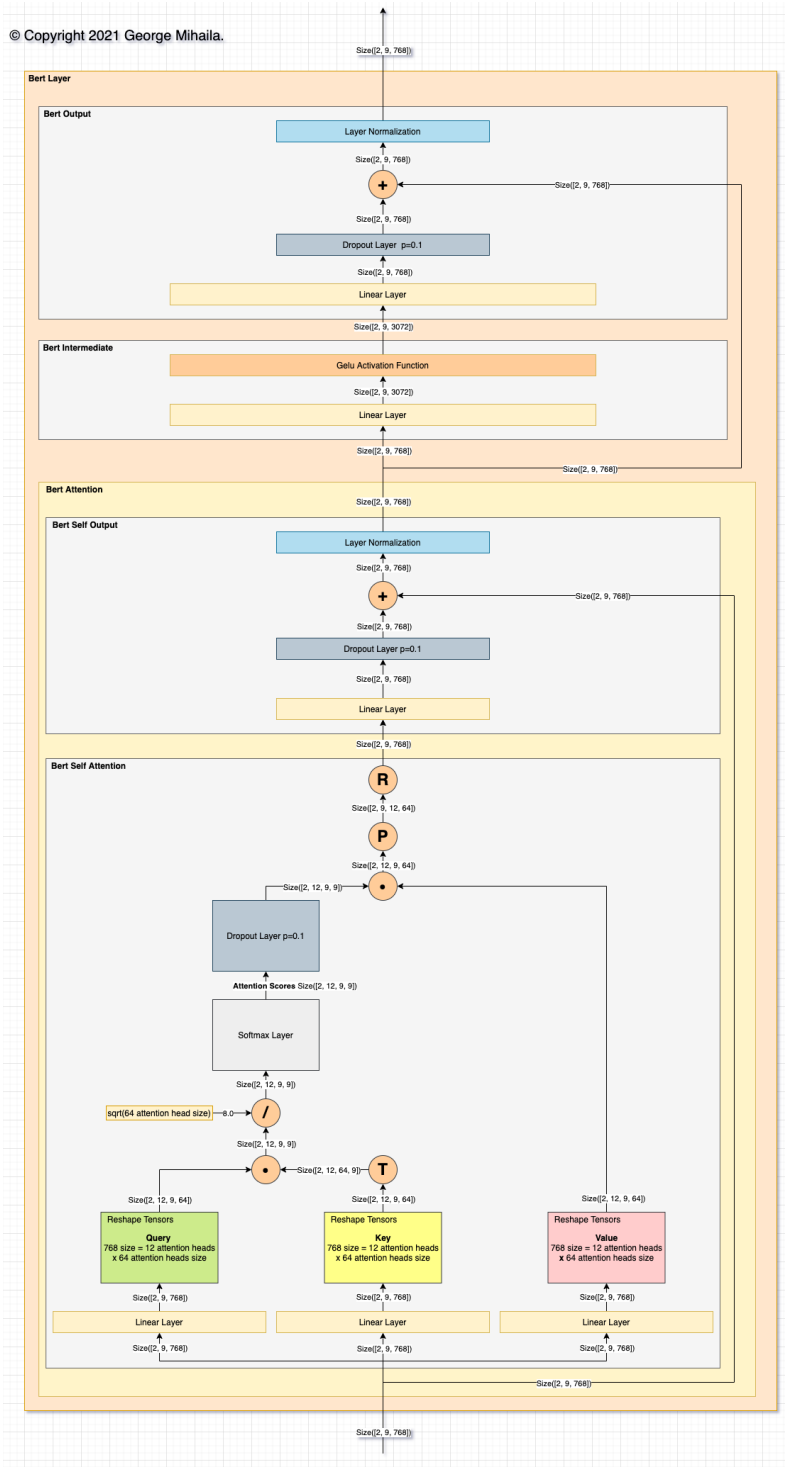


Figure 4.5: Encoder Module

Pooling Layer

Fig 4.6 shows that the encoder output has dimensions of (2, 9, 768). Each row corresponds to the enriched representation of a token from the input sentences. For semantic similarity task, we need a vector that captures the meaning of the entire input sentence. To achieve this, the pooling layer extracts [CLS] token, because it learns to encode the relationship between two sentences during the NSP task. The embedding of the [CLS] token is then passed through a linear layer followed by a non-linear tanh activation function.

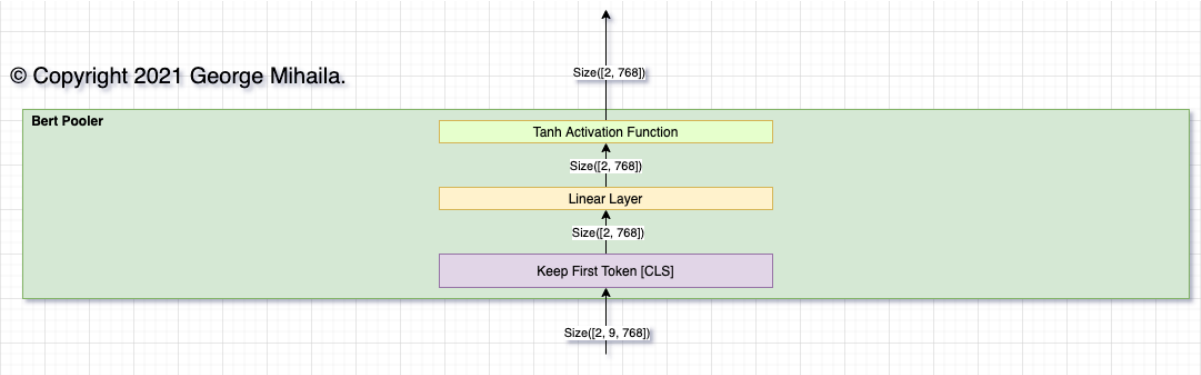


Figure 4.6: Pooling Layer

Classification Head

Finally, the output from the pooling layer is passed through the classification head, which projects the pooled embedding into a space with a dimensionality equal to the number of classes. In this project, we set the number of classes to 1 because we want to produce a score representing the similarity between the two input sentences. The activation layer used is the identity function, so the output is simply the logits. Since we are performing a re-ranking task, a score between 0 and 1 is not required.

4.2.2 Cross-Encoder Implementation Details

Training and Evaluation

Models are trained using the Tesco-Hybrid dataset across three distinct periods: weeks 38 to 41, weeks 40 to 43, and weeks 42 to 45. Each training sample consists of a tuple with three elements: product information of the original product, product information of the substitute product, and the ground-truth Bayesian acceptance rate collected during test week between them (examples of product information pairs are shown in Fig 5.5). The goal during training is to minimize

the Binary Cross-Entropy (BCE) loss, which measures the distance between two probability distributions. This loss function is particularly suitable because the Bayesian acceptance rate can be interpreted as a probability (probability of accepting the substitute). The BCE loss is defined as:

$$\text{BCE Loss} = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p))$$

where y is a continuous value between 0 and 1 representing the true probability that the substitution will be accepted (Bayesian acceptance rate), and p is the predicted probability that the substitution will be accepted.

Model evaluation occurs after each training epoch using a customised evaluator from the Sentence Transformer library, `CERerankingEvaluator`. It is originally designed for binary scores, we adapted it for continuous scores. The evaluation dataset, derived from Tesco-Offline for weeks 42, 44, and 45 – following each training period – consists of tuples of original and substitute product information. The model predicts similarity scores for these product information pairs, the pairs are ranked using the similarity scores and ranking quality is evaluated using the NDCG metric. The model that achieves the highest NDCG@10 score during training is saved.

Finally, the saved models are applied to the Tesco-Candidate dataset to predict scores for candidate product-substitution pairs. For consistency with the current Tesco implementation, candidate pairs without historical results are removed. These predicted score are used to rank the candidates, and their rankings are evaluated using NDCG and NDCG$_f$ metrics against actual product-substitution pairs in the test weeks. In practice, based on stock availability, only the highest ranked available substitutions from the model's recommendation are shown to offline store pickers, who then make the final decision on which product to substitute.

4.3 Chapter Summary

In this chapter, we present the design of a two-component recommendation system that integrates an LLM with RAG for feature enhancement and uses a cross-encoder for candidate re-ranking. The first component uses search engines such as Google for information retrieval, with `gpt-4o-mini` as the generative model; this enables the system to access up-to-date and additional product information, thereby improving the relevance of product information. We also present the architecture of the pre-trained `ms-marco-MiniLM-L-6-v2` model, detailing how its components – such as the special tokens, the multi-head self-attention layer, and the classification head – contribute to the cross-encoder's ability to understand both word-level and sentence-level meaning. Finally, we outline the implementation details, including the fine-tuning objective, and an overview of the training and evaluation process.

Chapter 5

Experiments

This chapter presents a series of experiments aimed at optimising and evaluating the performance of the proposed pipeline. We begin with an overview of the experimental design, including the evaluation metrics and baseline models used for comparison. Experiment 1 examines the impact of different prompting styles on performance, while Experiment 2 investigates how different inputs affect the performance of the cross-encoder. Experiment 3 focuses on hyperparameter tuning and analyzes the sensitivity of performance to these hyperparameters. For each experiment, we clearly define the objectives, describe the experimental setup, and analyse the results and their implications. Taken together, these experiments contribute to a deeper understanding of the recommendation system's behaviour and provide valuable insights into its practical applications and potential limitations.

5.1 Overview

This section presents the results of three experiments designed to explore different aspects of the trained models. Each model was trained for 30 epochs on the training dataset and then evaluated on the test dataset. Due to computational limitations, products from five divisions (F, T, B, W and N) are used in the experiment. In addition, comparisons are made with the baselines. The experiments are:

- **Experiment 1: Prompting Style** This experiment examines the effect of two different prompting styles – direct summarization and feature extraction – used for large language models on the performance of the recommendation system.
- **Experiment 2: Input to Cross-Encoder** This experiment examines the effect of different input product information on model performance, testing a total of four different combinations.
- **Experiment 3: Hyperparameter Tuning** In this experiment, key hyperparameters—batch size and learning rate—are systematically tuned to determine the optimal settings for cross-encoder performance in each division.

5.1.1 Evaluation Metrics

This section introduces the two evaluation metrics we will use throughout the three experiments, NDCG and NDCG$_f$.

Normalized Discounted Cumulative Gain (NDCG)

NDCG is a metric used to evaluate the quality of ranked lists generated by recommendation systems. It takes into account the position of relevant items in a ranked list and assigns higher scores to relevant items that appear earlier. NDCG is the normalised version of DCG and is calculated as follows

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k} \tag{5.1}$$

Where DCG is calculated as follows:

$$\text{DCG@}k = \sum_{i \in I} \frac{\text{rel}_i}{\log_2(i+1)} \tag{5.2}$$

where:

- rel_i is the relevance score of the product-substitution pair with ranking i. Note that i starts from 1 and I contains rankings of product-substitution pairs with ground truth scores (i.e. occurr in the test week). Since the final decision is made by offline store pickers, some product-substitution pairs may not appear in the test week.
- k is the rank position up to which the DCG is calculated. In this project, we use $k = 10$ as discussed in section 3.1.3.

And IDCG is the maximum possible DCG, calculated as:

$$\text{IDCG@}k = \sum_{i \in I} \frac{\text{rel}_i^*}{\log_2(i+1)} = \sum_{i \in I} \frac{1}{\log_2(i+1)} \tag{5.3}$$

Where rel_i^* is the relevance score at position i in the ideal ranked list. For this thesis, we will use $\text{rel}_i^* = 1, \forall i$, for consistency with value currently used by TESCO.

Normalized Discounted Cumulative Gain Fraction (NDCG_f)

NDCG_f is a metric designed to evaluate our model by considering the coverage of the predictions in the test week. NDCG_f is calculated using the following formula:

$$\text{NDCG}_f@k = \frac{\text{DCG}_f@k}{\text{IDCG}@k}$$

with:

$$\text{DCG}_f@k = \sum_{i=1}^{p} \frac{\delta_i}{\log_2(i+1)} \quad \text{and} \quad \text{IDCG}_f@k = \sum_{i=1}^{p} \frac{1}{\log_2(i+1)}$$

where $\delta_i = 1$ if the i-th product-substitution pair occurs in the test week and 0 otherwise.

NDCG_f is necessary due to the nature of the product substitution task. Our model will first predict substitutions for products within a candidate set. However, during the test week, not all product-substitution pairs may be included in the candidate set. This difference arises because the substitution decisions are eventually made by the pickers at Tesco, who act as the final checkers, so the rankings generated by the model are for reference only, as shown in Figure 5.1. The limitation of NDCG scores is that only those product-substitution pairs occur in the test week contribute to it, as shown in the Formula 5.1, 5.2 and 5.3. To address this, we introduce NDCG_f, a modified NDCG that evaluates both the coverage of the model's predictions over the test set and the reliability of the NDCG score.

More specifically, if the model scores high on NDCG but low on NDCG_f, this suggests that while some of the recommended substitutions perform well (as reflected by the high NDCG), many others do not occur in the test week. This could be because the products are new, or because the pickers reject them due to the high irrevalence of the recommendation, or because the pickers are biased towards the previous Dunnhumby solution. The problem of pickers selecting substitutes not recommended by the model leads to low compliance rates, which in turn leads to certain recommended substitutes not appearing during the test week, thus reducing the NDCG_f score. A low NDCG_f indicates the model is unreliable and recommends irrelevant substitutions, making it a crucial metric for evaluating both test week coverage and model reliability.

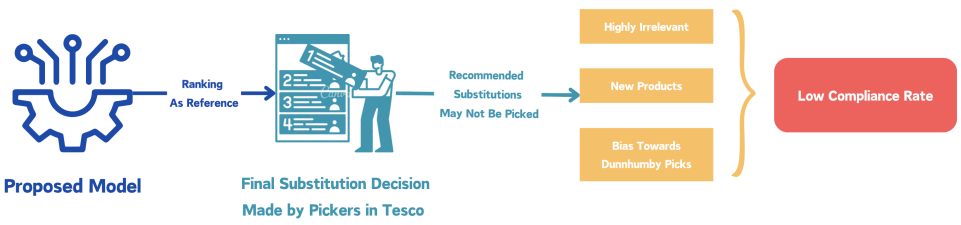


Figure 5.1: Illustration of Substitution Process and Low Compliance Rate

5.1.2 Baselines

We use two baseline methods for comparison: the Naive Method and the Bayesian Naive Method. We will explain them in detail in the following sections.

Naive Method

This model predicts the acceptance rate for a test week by directly using the historical Bayesian acceptance rate (details in Equation 3.1) from training weeks for each product-substitution pair. It assumes that past acceptance rates are a straightforward predictor of future acceptance rates.

Naive Bayesian Method

This method improves prediction by assuming a prior beta distribution, specifically a Beta(3,3) distribution, and updating this prior with the historical number of acceptances and rejections observed during the training weeks for each product-substitution pair. More specifically, let n represent the total number of picks, k the number of acceptances, and $n - k$ the number of rejections; the posterior distribution is then updated to Beta$(3 + k, 3 + n - k)$. The final predicted acceptance rate is derived from the 60% quantile of this posterior distribution. This approach is advantageous because

- The Beta(3,3) prior acts as a regulariser, helping to stabilise predictions in scenarios with limited historical data and remaining broad and uninformative.
- Integrating the Beta prior with empirical data (historical numbers of acceptances and rejections) allows the model to adaptively refine its estimates, leading to more reliable and accurate predictions.
- Using the 60% quantile as the prediction score provides a slightly optimistic estimate, reducing the risk of overestimation that might occur with more extreme quantile choices. This quantile choice has been validated through experimentation by the Tesco Data Science team, where they found that the 60% quantile gave the best performance.

This Bayesian approach combines prior knowledge with observed data, providing a probabilistic method of modelling acceptance while accounting for uncertainty.

5.2 Experiment 1: Prompting Style

5.2.1 Task Objectives and Description

The main objective of this experiment is to understand how different styles of prompts impact the summarized descriptions generated by LLMs and how these changes in LLM-augmented descriptions affect the quality of the final substitution candidate rankings.

Our experimental process involves feeding the title of a product and its web-scraped description into an LLM. The task of the LLM is to enrich and summarise this information. Figure 5.2 illustrates the prompts we used (only one example due to page limitations, in practice few-shot learning is used). The LLM first enriches the product information, and the output from this step is then used in two different summarisation approaches: direct summarisation and feature extraction followed by structured sentence construction. By varying the summarisation prompts, we can compare the LLM's ability to process the information, resulting in more accurate summaries that effectively capture the important features in different product divisions.

Initially, the LLM struggles to accurately extract the target features, often missing important details or producing redundant information. To overcome this, we use the technique of few-shot in-context learning. We also manually create customised prompts for each product category, recognising that different categories have unique characteristics. For example, the origin of a product is important in the alcoholic beverages category, but less relevant for snacks. The customised prompts for alcoholic products are shown in Figure 5.2.

5.2.2 Results

We will first present the results in Figure 5.3 along with key observations. We will then provide an analysis using actual examples of LLM output from different prompt styles, as shown in Figure 5.4.

Observation

Figure 5.3 compares two prompting styles, labeled as 0 for direct summarization and 1 for feature extraction, across different divisions (F, T, B, W and N) using the NDCG and NDCG$_f$ metrics. The cross-encoders used in this experiment are not trained to avoid the effect of different training dataset quality across divisions. The input to the cross-coder is feature2feature, as it incorporates the most information from the LLM output and should best reflect the effect of different prompting styles.

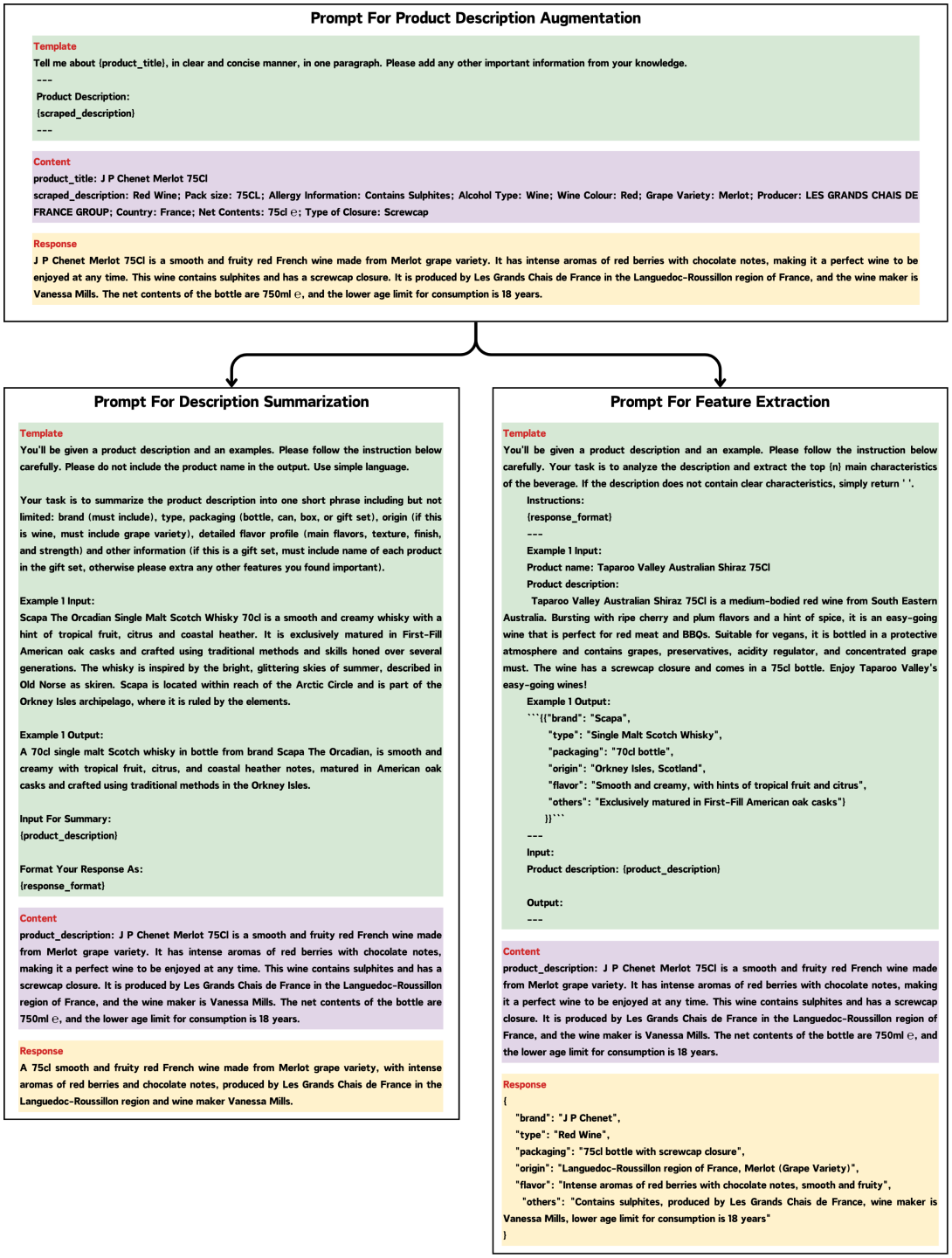


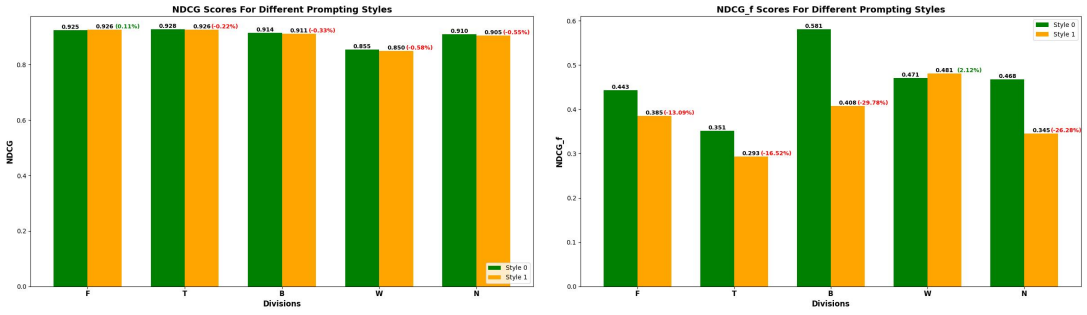
Figure 5.2: Example Prompts for Division W

The results show that prompt style 0 generally produces higher NDCG scores than prompt style 1, and this trend is consistent across four of the five divisions. The difference between the two styles in terms of NDCG performance is minimal, with percentage changes ranging from -0.55% to 0.11%. However, the results show a remarkable variance in NDCG_f scores, especially with the highest percentage change (-29.78%) observed in division B for prompt style 0 and the lowest percentage increase (2.12%) observed in division W for prompt style 0. This suggests that the prompt style has a greater influence on NDCG_f scores than on NDCG scores.

Analysis

The main differences between these two prompt styles lie in the output format and the level of freedom given to the LLMs. Prompt style 0 allows more freedom, enabling the LLMs to generate summarized descriptions with features presented in an order of their choice, including additional features they deem important. This flexibility is especially beneficial in Divisions T, B, W, and N, as shown in Figure 5.4, where the increased freedom results in more detailed, coherent, and useful information (highlighted in green). For instance, in describing *Swan Slim Loose Filters*, prompt style 0 explicitly mentions multiple color options, providing extra details that help the cross-encoder distinguish features more effectively.

In contrast, prompt style 1 restricts the output by using the LLMs as feature extractors and then arranging the extracted features into structured sentences. This approach is particularly effective for products in Division F, as shown in Figure 5.4. The structured output of prompt style 1 offers concise, standardized sentences with key features presented in a predefined order. For example, in the description of "Actimel Multifruit Immunity Live Yoghurt Drink Multipack 8X100g", prompt style 0 produces redundant and less useful information (highlighted in red), whereas prompt style 1 provides a clearer and more organised description that is easier for the cross-encoder to process and compare.



(a) NDCG Scores for Experiment 1 (b) NDCG_f Scores for Experiment 1

Figure 5.3: Comparison of model results for prompting styles: "0" is direct summarization, "1" is feature extraction. Percentage changes (Style 1 vs. Style 0) shown in parentheses, with decreases in red and increases in green.

Division	Title	Prompt Style 0	Prompt Style 1
T	Rizla Green Papers Multi 5 Pack	A pack of five green rolling papers for cigarettes from brand Rizla, intended for individuals 18 years or older and manufactured by Imperial Tobacco Limited in the EU. (Price: 1.63)	Rolling papers from Rizla with price 1.63, sized at 5 pack, packaged in pack of papers, and other features like green color, for rolling cigarettes, age restriction 18+.
	Swan Slim Loose Filters 165Pk	Filter tips from brand Swan, available in green, red, blue, silver, and liquorice, with 165 pieces per pack and each tip weighing 11.5g, intended for customers 18 years or older and with a full range of products including papers available on the website. (Price: 0.97)	Filter tips from Swan with price 0.97, sized at 165 pack, 11.5g each, and other features like slim loose filter tips, available in multiple colors, intended for customers 18+.
B	Tesco Chocolate Yule Log 260G	A 260g pack of chocolate-flavoured sponge roll with chocolate-flavoured buttercream, covered in milk chocolate and finished with a sweet dusting from Tesco, containing wheat flour, sugar, pasteurised egg, and whey powder (milk) among other ingredients, suitable for approximately 7 servings but may contain peanuts and nuts. (Price: 2.18)	Chocolate yule log from Tesco with price 2.18, sized at 260g, packaged in pack, with a flavor of chocolate, which contains wheat flour, pasteurised egg, and whey powder (milk). May contain peanuts and nuts.
	Tesco Finest Triple Chocolate Shortbread 4 Pack	A 4 pack of all-butter shortbread slices with milk, dark, and white Belgian chocolate chunks from Tesco Finest, perfect for indulging with tea or coffee but may contain allergens such as peanuts, nuts, milk, sesame, egg, and soya. (Price: 1.64)	Shortbread from Tesco with price 1.64, sized at 4 Pack, packaged in pack, with a flavor of triple chocolate, which contains wheat, milk, and soya. May contain peanuts, nuts, sesame, and egg.
N	Shockwaves Curls & Waves Mousse 200Ml	A 200ml hair styling mousse from Shockwaves that defines curls and waves, provides flexible hold, and reduces frizz with ingredients like PVP, VP/VA Copolymer, and Chitosan. It can be applied easily by shaking well and holding the can upside down before scrunching it into damp hair, and can be left to dry naturally or blow-dried with a diffuser. (Price: 2.69)	Curls & Waves Mousse from Shockwaves, sized at 200ml, which includes Defines curls and waves, provides flexible hold and anti-frizz effect. Formulated with PVP, VP/VA Copolymer, and Chitosan. Easy to apply by shaking well and holding the can upside down before scrunching it into damp hair. Can be left to dry naturally or blow-dried with a diffuser. Leaves hair feeling smooth and bouncy with a pleasant fragrance..
F	Actimel Multifruit Immunity Live Yoghurt Drink Multipack 8X100g	A gluten-free yogurt drink from Actimel with L.Casei Danone cultures, vitamins B6 and D, and multifruit flavors of pineapple, peach, orange, and strawberry, containing 10 billion live L. casei cultures, suitable for vegetarians, and with 1/3 of the daily reference intake (RI) of vitamin D and 15% of the RI for vitamin B6, packaged in recyclable bottles and cardboard. (Price: 3.13)	Multifruit Immunity Live Yoghurt Drink from Actimel with price 3.128, sized at 8x100g, packaged in Recyclable plastic bottles, with a flavor of Pineapple, Peach, Orange, Strawberry, which is Gluten-free, Vegetarian, and requires Ready-to-drink preparation.
W	Mateus Rose Wine 75Cl Plus 33% Extra Free	A 75cl bottle of youthful and fresh Mateus Rose wine with a touch of sparkle and lovely hints of red fruit, perfect for any occasion and pairs well with seafood, salads, and light pasta dishes. (Price: 5.22)	Rose Wine with price 5.22 pounds from the brand Mateus, packaged in 75cl bottle with 33% extra free, and is characterized by a flavor of Youthful and fresh with a touch of sparkle, hints of red fruit, and a slightly sweet taste and it is also Pairs well with seafood, salads, and light pasta dishes.
	Greene King Icebreaker Pale Ale 4X440ml	A pack of 4 x 440ml unfiltered pale ale cans from Greene King Icebreaker, brewed with Citra and Simcoe hops, featuring fruity hop aroma, citrus and tropical fruit flavors, balanced with soft malty tones, and perfect for pairing with fish and chips, juicy burgers, and pizzas. Winner of a 3-star Great Taste Award in 2023, contains malted barley and wheat, and best served chilled from the fridge. (Price: 5.65)	Pale Ale with price 5.65 pounds from the brand Greene King, packaged in 4 x 440ml can pack, and is characterized by a flavor of Citrus and tropical fruit flavors, balanced with soft malty tones and it is also Unfiltered, brewed with Citra and Simcoe hops, winner of a 3-star Great Taste Award in 2023.

Figure 5.4: Examples for Comparing Different Prompting Styles: Prompting Style 0 represents Direct Summarization, while Prompt Style 1 involves Feature Extraction followed by structuring into a sentence. Words added to form structured sentences in Prompt Style 1 are highlighted in purple. Information that is more effectively presented in one style is marked in green, whereas less effective presentation is marked in red.

5.3 Experiment 2: Input to Cross-Encoder

5.3.1 Task Objectives and Description

This experiment explores the impact of different text inputs on model performance. The primary goal is to understand how variations in the input text influence the language model's effectiveness in predicting the ranking of candidate substitutes.

We want to investigate that whether the type of textual data used to train the cross-encoder can greatly impacts its performance. Therefore, we explore four different combinations of input features, as shown in Fig 5.5, including:

- **Title to Title (title2title):** Original product title against substitute product title.
- **Title to Features (title2feature):** Original product title against the summarized features of substitute product generated by large language models.
- **Features to Features (feature2feature):** The summarized features of the original product against the summarized features of the substitute product.
- **All to All (all2all):** The original product title and its summarized features against the substitute product title and its summarized features.

Figure 5.5: Input Combinations Tested for Cross-Encoder Training

5.3.2 Results

In this section, each cross-encoder was trained with parameters: 30 epochs, a learning rate of 2×10^{-5}, a batch size of 32, and a prompt style selected as the most effective from Experiment 1. The experiments spanned five divisions, specifically F, T, B, W, and N, and were conducted in test weeks 42, 44, and 46. Figures 5.6 and 5.7 illustrate the average NDCG and NDCG$_f$ scores across the three test weeks, comparing cross-encoders against the baselines. Percentage changes in performance metrics relative to the stronger baseline are shown in brackets (red for decrease, green for increase), with the best-performing cross-encoder in each division highlighted in bold. For a detailed breakdown of results for each division in each test week, please refer to Table A.1.

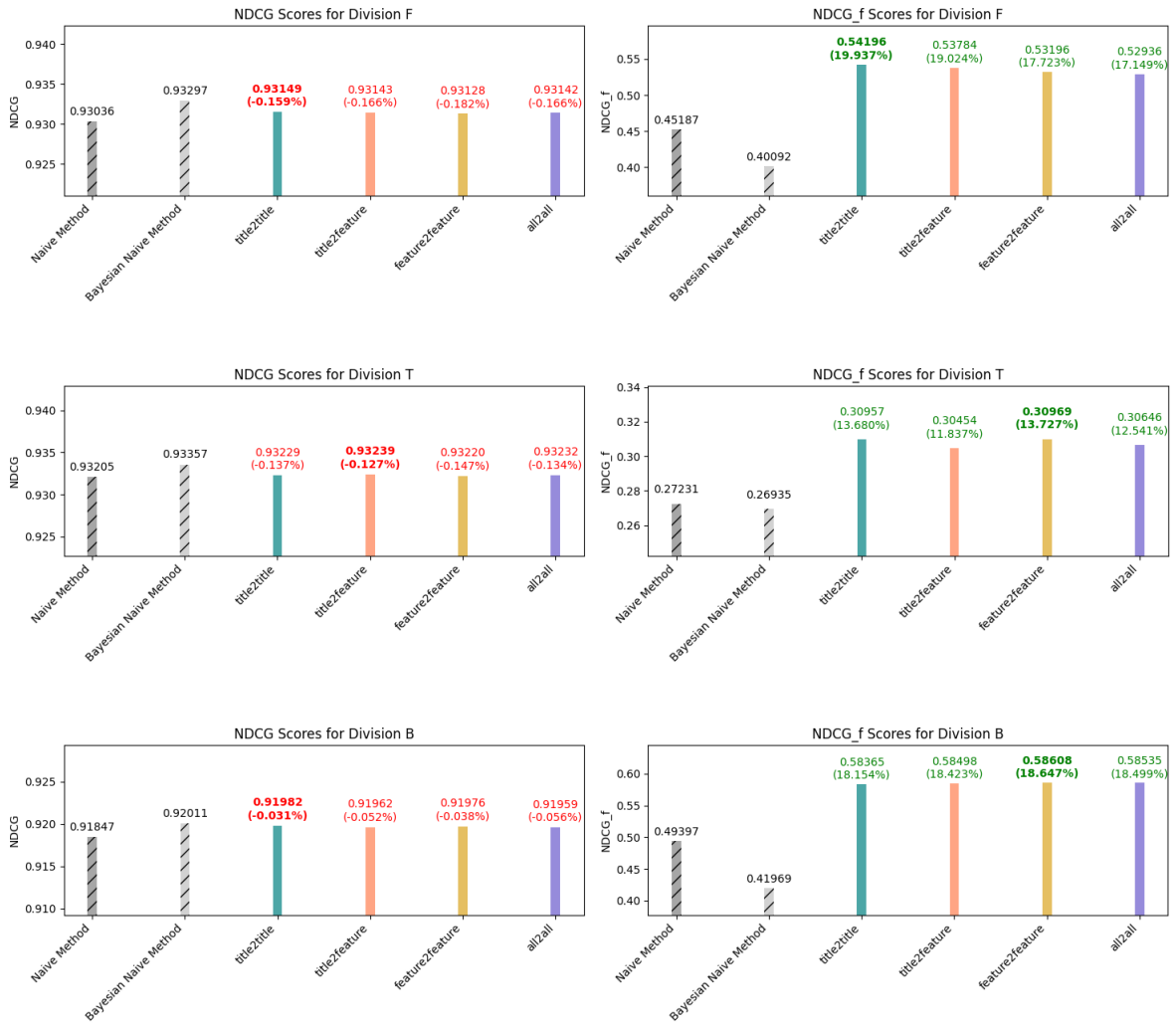


Figure 5.6: Comparison between Baselines and Cross-Encoders for Division F, T and B

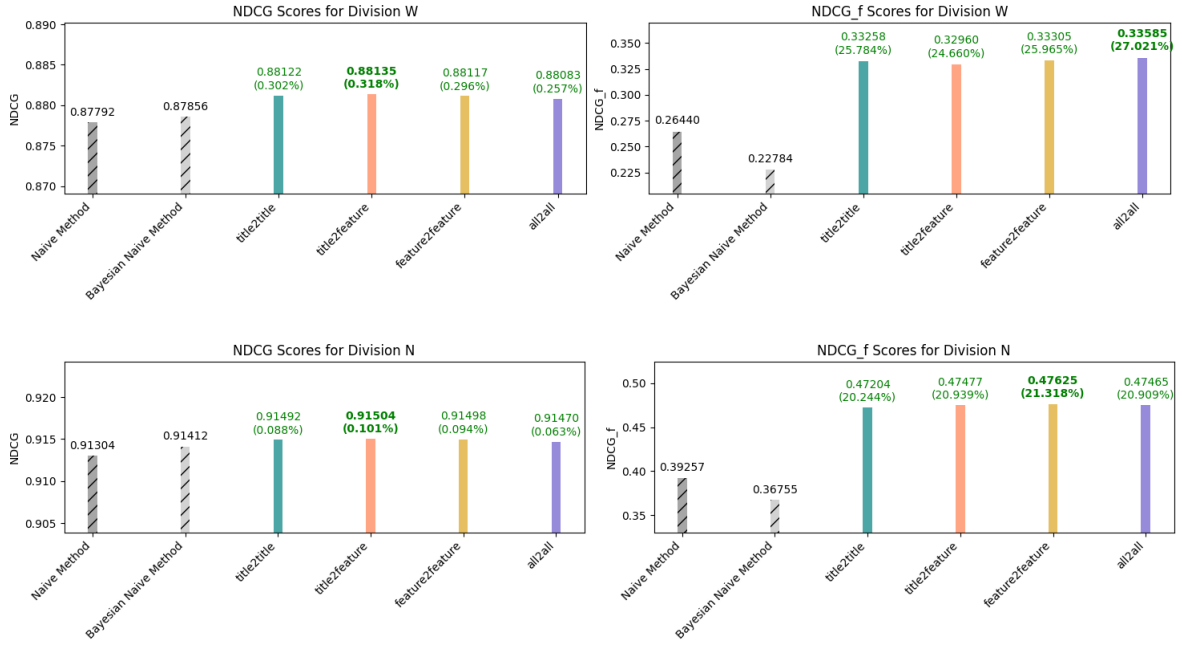


Figure 5.7: Comparison between Baselines and Cross-Encoders for Division W and N

Observation

The Naive Method has lower average NDCG scores across all divisions, indicating that it is the weaker baseline in terms of ranking quality. In contrast, although the Bayesian Naive (BN) Method achieves higher ranking quality, it has lower NDCG_f, implying greater uncertainty in its performance. This uncertainty arises because the BN Method tends to rank more substitutions that do not occur in the test week higher, reflecting its slightly optimistic modelling of historical acceptance rates compared to the purely historical approach of the Naive Method.

In Divisions F, T, and B, cross-encoders achieve higher NDCG scores than the Naive Method but perform worse than the Bayesian Naive Method, unexpectedly. This leads to further analysis of the training set size (Table 5.1) and the input product information used for training the cross-encoders (Figure 5.8). However, in Divisions W and N, cross-encoders outperform both baselines, aligning with our expectations. We conduct further investigation into the reasons for this improved performance which involves analyzing the characteristics of products and their associated product information.

We observed consistent improvement in NDCG_f scores with cross-encoders, regardless of the textual information used in training. This matches our expectation as NLP-based methods can interpret product information and automatically filter out semantically irrelevant substitutions.

Incorporating LLM-augmented product information into the training of cross-encoders (ti-

tle2feature, feature2feature, and all2all) led to declines in NDCG scores in Divisions F and B, and a decrease in $NDCG_f$ score in Division F. We analyzed the product characteristics contributing to these outcomes in the subsequent section. Moreover, using more LLM-augmented product information (feature2feature and all2all) resulted in lower NDCG scores compared to using less (title2feature) across Divisions T, W, and N. This suggests that maintaining simpler original product information when compared to more detailed substitutes is the most effective configuration.

Analysis for Divisions F, T and B

Cross-encoders consistently outperform Naive Method but underperform relative to the BN Method in all divisions, regardless of product information used for training. The performance gap is most pronounced in Division F, with a decline ranging from -0.182% to -0.159%. This is followed by Division T, with a decline ranging from -0.147% to -0.127%, and then Division B, with a decline ranging from -0.056% to -0.031%. We will further investigate the reasons for these performance differences in more detail.

In the FRESH FOODS (F) and BREAD/BAKERY & REST (B) divisions, the diversity of consumer preferences make NLP-based methods less appropriate. To be more specific, choices in fresh food are deeply influenced by personal factors like taste preferences, dietary restrictions, and health considerations. These choices are highly individualized and cannot be adequately learned from just product titles or descriptions. Instead, they are more accurately predicted using historical purchase data, which the BN Method, with its probabilistic approach, is more effectively to capture underlying consumer behavior patterns.

In TOBACCO KIOSKS (T) division, where the amount of training data available is extremely limited, the complexity of the cross-coder could lead to overfitting. As detailed in Table 5.1, division T has a significantly smaller training set than other divisions. A dataset of around 3000 samples is insufficient to fine-tune a cross-encoder with millions of parameters.

Table 5.1: Size of Training Set for Each Division

Division	Mean ± Std
B	23335 ± 592.29
F	103438 ± 2412.97
N	85932 ± 2777.54
T	2822 ± 376.26
W	17431 ± 1105.85

We observed a performance decline in Divisions F and B after incorporating LLM-augmented product information into the cross-encoder training. Given the simplicity of the products in these divisions, the original titles were already informative enough. Forcing LLMs to augment the descriptions introduces unnecessary complexity and noise. Figure 5.8 illustrates the problems by comparing the original titles with the LLM-augmented descriptions. For example, the description for *parsley bunches* contains vague statements such as "a good source of vitamins and minerals" and "can be used in a variety of dishes", and the note "can be kept in the fridge for a few days" lacks key product insights to distinguish it from other substitutes. Similarly, the LLM-augmented description for *Kingsmill Soft White Medium 800g* unnecessarily lists all the ingredients, complicating the cross-encoder comparison, as it needs to evaluate all the specific ingredients rather than focusing on more relevant product characteristics.

Analysis for Divisions N and W

Cross-encoders generally outperform baselines in the WINES SPIRITS (W) and NON-FOODS GROCERY (N) divisions because customers evaluate these products based on more standardized features. For example, consumers evaluate wines using criteria such as brand, grape variety, and flavor profile. Similarly, for products in Division N, such as hair dyes and tablets, people consider functionality, effectiveness, safety, and compatibility with personal health conditions when making decisions. Cross-encoders, as transformer-based models with attention mechanisms, can learn consumer preferences and adjust attention to relevant features accordly by fine-tuning on historical data, resulting in more human-like decision-making and therefore, more accurate recommendations.

Cross-encoders trained on LLM-summarized descriptions (title2feature and feature2feature) outperformed those trained on raw titles alone (title2title). This is because LLM-augmented descriptions provide richer, more detailed information. For example, as shown in Figure 5.8, the summary description for *Jerome Russell B Blonde Max Cream Peroxide* includes useful details like personal conditions "suitable for light to dark brown hair" and effectiveness information "can lift up to 8-9 shades". Another reason is that raw product titles can be very informative in these divisions. For instance, *J P Chenet Merlot 75Cl* alone does not provide any key characteristic of the product. Through our feature augmentation strategy by using product information scraped from the web and the LLM's internal knowledge base, the product description is enriched with details such as "intense aromas of red berries and chocolate notes", as shown in Figure 5.8. This enhanced understanding allows the cross encoders to make more informed decisions, thereby improving their performance in Divisions W and N.

While cross-encoders trained with raw titles outperform in Division F and B, these tend to be for

simpler products where extra context offers less value. For more complex products, the LLM-augmented approach consistently performs better. Overall, we believe that these individual cases do not undermine the general utility of our feature augmentation strategy, but rather reflect the limits of complexity at which additional information is required.

Division	Title	Summary Description from LLM
F	Parsley Bunches 100G	Fresh parsley bunches from an unspecified brand, a good source of vitamins and minerals, versatile herb that can be used in various dishes, and can be stored in the refrigerator for a few days.
	City Kitchen Chicken Katsu Curry 380G	A convenient and easy-to-prepare meal from City Kitchen, Chicken Katsu Curry 380G features mild marinated chicken breast in a sweet curry sauce flavored with aromatic spices, suitable for those who are looking for a quick and tasty meal option.
T	Lambert & Butler Bright 100 Pack	A cigarette product from brand Lambert & Butler Bright, manufactured by Imperial Tobacco Limited in the EU with a lower age limit of 18 years for purchasing.
	Kensitas Club Rolling Tobacco & Cigarette Paper 30G	A 30G pack of rolling tobacco and cigarette paper from popular brand Kensitas Club, available only to individuals aged 18 and older.
B	Mr Kipling Cherry Bakewells 6Pk+2 Extra Free	A pack of 6+2 cherry bakewells with fruity plum and raspberry jam, moist almond sponge, soft icing, and a cherry on top from Mr Kipling.
	Kingsmill Soft White Medium 800g	An 800g pack of white bread with 20 slices (including crusts) from Kingsmill, suitable for vegetarians and vegans, and certified Kosher and Halal. It contains wheat flour, water, yeast, salt, vegetable oils, soya flour, vinegar, calcium propionate, emulsifier, and ascorbic acid, and may contain cereals containing gluten.
W	J P Chenet Merlot 75Cl	A 75cl smooth and fruity red French wine made from Merlot grape variety, with intense aromas of red berries and chocolate notes, produced by Les Grands Chais de France in the Languedoc-Roussillon region and wine maker Vanessa Mills.
	Blaumeister Hock 75Cl	A 75cl bottle of medium sweet and fruity white wine from Germany, Blaumeister Hock, with exotic fruit flavors, best served chilled with desserts or on its own, providing 6 servings.
	Smirnoff Ice Vodka Mixed Drink Bottle 4% Vol 70cl	A 70cl bottle of pre-mixed Smirnoff No.21 vodka and lemon-lime soda with a crisp, citrus taste, best served over ice with a lemon wedge. Perfect for gifting to vodka enthusiasts.
N	Jerome Russell B Blonde Max Cream Peroxide 40 Volume	A 75ml bottle of maximum cream peroxide from Jerome Russell B Blonde, designed for use with Bblonde High Lift Powder Bleach, suitable for light to dark brown hair and can lift up to 8-9 shades, made in England and includes hydrogen peroxide, cetyl alcohol, avocado oil, and seaweed extract.
	Xls Medical Max Strength 40 Tablets	A weight management supplement from XLS-Medical Max Strength that reduces calorie intake from carbohydrates, sugar, and fat with Clavitanol™, clinically proven and certified as a medical device for effective weight management when taken with a balanced diet and regular physical activity, and helps lower blood glucose and insulin levels.

Figure 5.8: Comparison of Product Titles and LLM Summarized Descriptions by Division: Green phrases indicate useful information, while red ones are repetitive or useless.

5.3.3 Ablation Study: Finetuning Effect

Fine-tuning cross-encoders on the Tesco-Hybrid dataset is both computationally and time-intensive. Therefore, it is essential to evaluate its impact on model performance. This section presents an ablation study of fine-tuning effects, with averaged results in Figure 5.9 and more detailed results for each week in Table A.2.

Observation

Fine-tuning the cross-encoders on the Tesco-Hybrid dataset improves average performance across the three test weeks in all divisions. NDCG scores increase by 0.0363% to 0.921%, while NDCG_f scores increase more significantly, ranging from 1.17% to 21%.

However, Division T shows a performance decline after fine-tuning in weeks 44 and 46, as shown in Table A.2. This issue likely arises from the significantly smaller training dataset in this division, as indicated in Table 5.1. With around 3,000 samples, the complex cross-encoder model (with millions of parameters) may overfit to noise rather than learning meaningful patterns. The slight performance drop in cross-encoders trained with LLM-augmented descriptions suggests that using raw titles is more effective for this division.

Training cross-encoders with LLM-augmented product information resulted in the highest NDCG increase in 4 out of 5 divisions, and an NDCG_f increase across all divisions. However, the model with the largest percentage increase after fine-tuning does not necessarily have the highest score. For example, the all2all model in Division F leads to the highest NDCG percentage increase while the title2tile model has the highest NDCG score. Although LLM augmentation can improve results, when dealing with straightforward products, the simplicity and clarity of raw titles prove more effective.

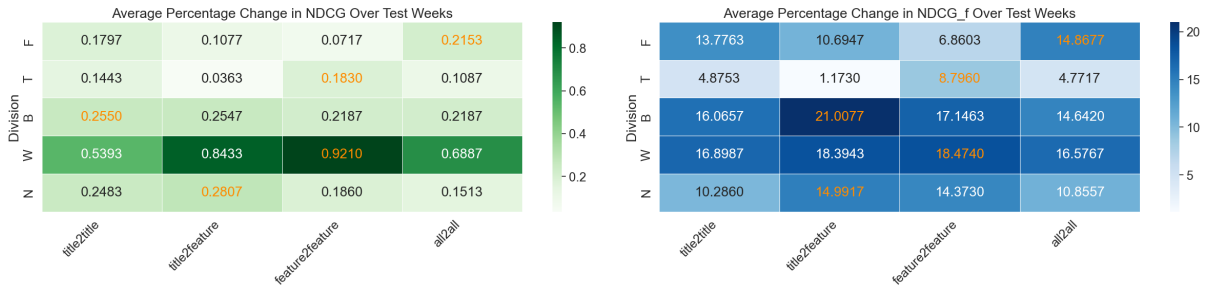


Figure 5.9: Averaged percentage changes in NDCG and NDCG_f between non-finetuned and finetuned models. Darker shades indicate larger percentage changes and the highest average percentage increase for each division is highlighted in orange.

5.4 Experiment 3: Hyper-parameter tuning

5.4.1 Task Objectives and Description

Having identified the optimal prompt style and input information configurations for the cross-encoder from Experiments 1 and 2, we are now ready to perform hyperparameter tuning. In this section, we aim to improve model performance by identifying the best set of hyperparameter configurations and understanding their impact on the results. In particular, we will focus on two crucial hyperparameters: learning rate and batch size.

Due to computational limitations, we only evaluate two settings for learning rate and batch size – specifically, a higher and a lower value for each parameter. We explore all four possible combinations of these values. The parameters are as follows:

- Learning Rate : $[2 \times 10^{-3}, 2 \times 10^{-5}]$
- Batch Size: [32,128]
- Cross-Encoder Input: title2title for Divisions F and B, title2feature for Divisions T, W, and N.
- Prompting Style: 0 for Division T and N, 1 for Division W.

The optimal configurations for prompting style and input information identified in Experiments 1 and 2 are used. It should be noted that, as cross-encoders are not trained with LLM-augmented information in Division F and B, there is no need to consider prompting style for them.

5.4.2 Results

Figure 5.10 shows heatmaps of the averaged NDCG and NDCG_f scores with standard deviations across all five divisions and four hyperparameter sets. A detailed weekly breakdown of the results can be found in Figure A.1. In the heatmaps, darker shades indicate higher scores.

For Divisions F and B, the NDCG scores tend to increase as the learning rate decreases and the batch size increases (shown by darker shades from top right to bottom left in Figure 5.10), while for Division T the opposite trend is observed. In these three divisions, the models show low sensitivity to these parameters, as shown by the closely clustered lines in Figure A.1. The limited differences in the performance of different hyperparameter configurations for Divisions F, T and B may be due to the inability of the model to adequately capture the product characteristics in their datasets. This suggests that adjusting the hyperparameters may not significantly improve performance as the model, a complex pre-trained cross-encoder, may already be operating at its limits in this context. However, this may also be due to the very limited hyperparameter space we explored. If we had more time, further experiments could explore this in more depth.

For Divisions W and N, the model captures the product characteristics well, which is reflected in higher performances than the baselines, as shown in Figure A.1. In particular, a lower learning rate improves performance, which matches our expectations, as lower learning rates can prevent the training process from overshooting the minima of the loss function, thus helping the model to converge more smoothly and stably to the optimal weights. Among the models with lower learning rates, the one with a larger batch size performs better, meaning that less noise is preferred in the computation of the gradient. Although the batch size did not have much effect on the NDCG values, it did result in a much higher memory consumption, as a larger batch size means that more data is processed simultaneously. An out-of-memory problem occurred during training, which was solved by reducing the number of models trained in parallel.

The trends in NDCG_f scores are more consistent across all divisions, with models using lower learning rates generally performing better, as shown by the darker shades in the right half of the heat maps in Figure 5.10. Interestingly, while the NDCG_f scores show clearer improvements, the variability between weeks is greater, as indicated by the standard deviations being more than 10 times larger.

To conclude, the parameters for each division that achieved the highest NDCG score are presented in Table 5.2. However, for practical use at Tesco, it may not be feasible to use different parameters for each department. Based on our experiment, a smaller learning rate is recommended. The choice of batch size depends on the needs of the company; while a larger batch size speeds up the computation per epoch, it may cause out-of-memory problems.

Division	Prompting Style	Product Information Type	Learning Rate	Batch Size
F	/	title2title	2e-5	128
T	0	title2feature	2e-3	32
B	/	title2title	2e-5	128
W	1	title2feature	2e-5	32
N	0	title2feature	2e-5	32

Table 5.2: Parameters for Each Division that Achieve Highest NDCG Score

5.5 Chapter Summary

This chapter presents three experiments that analyse the impact of prompt style, product information type, and hyperparameter values on recommendation system performance. By optimising these factors, we achieve incremental improvements, eventually outperforming the baseline in categories with complex product information, such as alcoholic beverages and non-food groceries, as shown by improved NDCG scores, ranging from 0.04% to 0.73%. Our method significantly increases NDCG_f compared to the naive baseline in all divisions, ranging from 6.86% to 32.51%, indicating significant improvement in model reliability and recommendation relevance, which also speeds up the substitution process in offline settings.

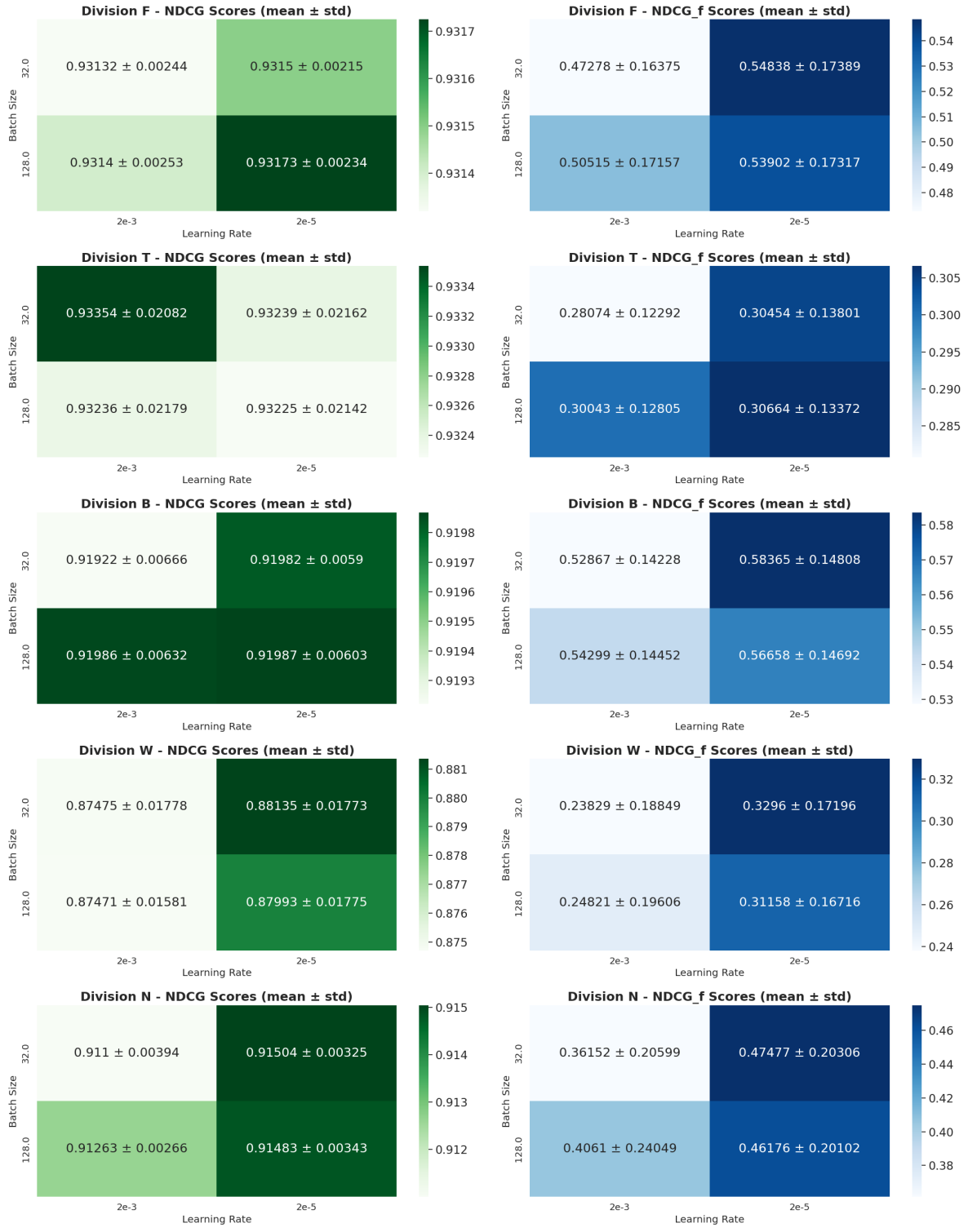


Figure 5.10: Heat maps showing NDCG and NDCG$_f$ scores across divisions for different hyperparameters, with shading based on the mean values of NDCG and NDCG$_f$.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis explores the use of Large Language Models for the task of offline product substitution, specifically by utilizing an LLM with Retrieval-Augmented Generation for feature augmentation and fine-tuning cross-encoders to re-rank substitute candidates. Incorporating LLMs into recommendation systems is a relatively novel approach. To the best of our knowledge, this is the first research focused on using LLMs for product substitution. We developed a recommendation system and conducted extensive experiments to determine whether a purely NLP-based method could accurately reflect consumers' dynamic preferences for Tesco products. Using OpenAI's `gpt-4o-mini` model and the `ms-marco-MiniLM-L-6-v2` model, which is based on Google's `BERTForSequenceClassification`—a variant of BERT—the aim is to create a recommendation system capable of fully interpreting products and updating its knowledge with open-world information, creating a ranked list of product substitutes and generating a similarity score for future integration with other models.

Our recommendation system is a two-component system consisting of an LLM with RAG and a cross-encoder. The pipeline starts with a web scraper that retrieves up-to-date product information from Tesco's official website, which is processed by the LLM (`gpt-4o-mini`) to first augment it with its open-world knowledge and then produce a concise summary. For the cross-coder, we use the `ms-marco-MiniLM-L-6-v2` model, which is pre-trained on the MS MARCO dataset, a large dataset used for training models on passage ranking. This cross-encoder is built upon the BERTForSequenceClassification architecture, which has advanced bi-directional language representations and is designed to analyse the relationship between two sentences. This enhances the cross-encoder's ability to capture relationships between product sentences and generate accurate similarity scores for ranking purposes. Furthermore, it is fine-tuned on the Tesco-Hybrid dataset using Binary Cross-Entropy Loss and evaluated using NDCG and NDCG$_f$ metrics. Technologically, we use Azure OpenAI to interact with the LLM and Hugging Face's Sentence Transformer library to create the cross-encoder. Our results show the effectiveness of the recommendation system in achieving all three research objectives:

- Reframed the Substitution Task as an NLP Problem: We successfully reformulated the product substitution task as an NLP problem, using pre-trained language models to un-

derstand, analyse and compare complex product descriptions for better substitution recommendations.

- Proposed a New Recommendation System Integrating LLMs with Cross-Encoder: Our innovative approach combines the strengths of the LLM and retrieval-augmented generation with the cross-encoder. This powerful integration results in a highly effective system tailored to address product substitution task across multiple product categories.
- Conducted Extensive Experiments and Ablation Studies on TESCO Datasets: We conducted a series of extensive experiments on the Tesco datasets to investigate the influence of prompt design, product information and hyperparameters on system performance. These studies validated our approach and led to significant improvements in performance metrics, approximately 1% increase in NDCG and 30% increase in NDCG_f. This can help to significantly speed up the substitution process while increasing substitution accuracy.

This research demonstrates that LLMs are effective for product substitution tasks, providing a promising alternative to traditional methods. Our approach utilizes product information that is readily available from the retailer, which is often more practical than relying solely on customer behaviour inferred from historical data. While historical behavioural data can suggest substitution relationships, these indirect signals are not always reliable indicators. Additionally, our method emulates a more human-like decision-making process by learning from and mimicking how consumers evaluate product features. This has led to improvements in categories with complex product information, such as alcoholic beverages and non-food groceries, where traditional models often fail to capture the nuanced decision-making processes based only on historical data.

Despite the promising results, the models are parameter-heavy and expensive in terms of both training and inference costs. Specifically, training the cost encoder on the provided subset of the Tesco-Hybrid dataset takes about 7 hours, with inference taking about 1 hour on an NVIDIA Tesla T4 GPU. Although the need for speed in an offline setting is less strict than in an online setting—requiring weekly training and inference—the process is still significantly slower than traditional models. Therefore, for practical application in the future, efforts should be directed towards improving the model to achieve faster training and inference.

6.2 Future Work

In this section, we outline potential paths for future work aimed at enhancing the efficiency and performance of our recommendation system. Our focus will be on reducing training and inference time, improving prompt engineering process, and integrating with other models.

- **Efficient Model Training and Compression Techniques:** To reduce model training and inference time, we consider two advances: Parameter-Efficient Fine-Tuning (PEFT) for cross-encoders and model compression. PEFT allows large language models to be fine-tuned by selectively updating parameters, thereby conserving resources. This includes the use of adapters, small neural network modules inserted between layers of a pre-trained model, where only the adapter parameters are updated during fine-tuning, while the original model parameters remain unchanged. And Low-Rank Adaptation (LoRA), which approximates changes in the model's weight matrices with a low-rank matrix to maintain efficiency. In addition for compression techniques, quantization converts models to lower precision formats, such as 8-bit integers from 32-bit floats, reducing memory requirements and speeding up inference.
- **Streamlining the Prompt Engineering Process:** Currently, our project involves manually customising prompts for each product division, a process that is both time consuming and potentially less effective at capturing key product features. To address this, we can implement soft prompting techniques where learnable tensors are concatenated with input embeddings and optimised for specific datasets. This will reduce the need for extensive manual customisation and improve the ability of the model to adapt to new data autonomously.
- **Integrating Bayesian Models for Improved Predictions:** In our future work, we aim to enhance the predictive capabilities of our models by incorporating Bayesian models. Currently, our model has difficulty capturing personal customer preferences in certain categories, where Bayesian models have shown promising results in our previous experiments. However, Bayesian models face issues such as the cold-start problem, where reliance on historical data limits their initial effectiveness when data is limited, and challenges in fully interpreting product information. To overcome these limitations, we are considering incorporating similarity scores between products generated by our models into the prior distribution of the Bayesian model. These similarity scores will help the model to make informed predictions when historical data is insufficient. In addition, when there is sufficient observed data, the model can update the posterior distribution to more accurately reflect actual customer behaviour. By leveraging the strengths of both methods, this could lead to a more robust model.

In summary, the future work outlined aims to significantly advance the capabilities of our models through innovative techniques in model training, prompt engineering and integration with Bayesian approaches. By focusing on these strategic areas, we expect to improve not only the efficiency and accuracy of our models, but also their effectiveness in real-world applications.

References

[1] M. Zhang and J. Bockstedt, "Complements and substitutes in product recommendations: The differential effects on consumers' willingness-to-pay," *CEUR Workshop Proceedings*, vol. 1679, pp. 36–43, 2016. Publisher Copyright: © 2016, CEUR-WS. All rights reserved.; 2016 Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, IntRS 2016 ; Conference date: 16-09-2016.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics*, 2019.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. teusz Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *ArXiv*, vol. abs/2005.14165, 2020.

[4] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *ArXiv*, vol. abs/2302.13971, 2023.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[7] X. Liang, H. Wang, Y. Wang, S. Song, J. Yang, S. Niu, J. Hu, D. Liu, S. Yao, F. Xiong, and Z. Li, "Controllable text generation for large language models: A survey," 2024.

[8] M. Du, F. He, N. Zou, D. Tao, and X. Hu, "Shortcut learning of large language models in natural language understanding," *Communications of the ACM*, vol. 67, pp. 110 – 120, 2022.

[9] W. Zhu, H. Liu, Q. Dong, J. Xu, L. Kong, J. Chen, L. Li, and S. Huang, "Multilingual machine translation with large language models: Empirical results and analysis," *ArXiv*, vol. abs/2304.04675, 2023.

[10] C. E. Shannon, "Prediction and entropy of printed english," *The Bell System Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[12] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, pp. 179–211, 1990.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] C. Olah, "Understanding lstm networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[15] A. Jain, A. Rouhe, S.-A. Grönroos, and M. Kurimo, "Finnish language modeling with deep transformer models," *ArXiv*, vol. abs/2003.11562, 2020.

[16] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[17] B. Smith and G. Linden, "Two decades of recommender systems at amazon.com," *IEEE Internet Computing*, vol. 21, no. 3, pp. 12–18, 2017.

[18] J. Bennett and S. Lanning, "The netflix prize," 2007.

[19] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The youtube video recommendation system," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, (New York, NY, USA), p. 293â296, Association for Computing Machinery, 2010.

[20] M. De Nadai, F. Fabbri, P. Gigioli, A. Wang, A. Li, F. Silvestri, L. Kim, S. Lin, V. Radosavljevic, S. Ghael, D. Nyhan, H. Bouchard, M. Lalmas, and A. Damianou, "Personalized audiobook recommendations at spotify through graph neural networks," in *Companion Proceedings of the ACM on Web Conference 2024*, WWW '24, (New York, NY, USA), p. 403â412, Association for Computing Machinery, 2024.

[21] R. van Meteren, "Using content-based filtering for recommendation," 2000.

[22] C. C. Aggarwal, *Content-Based Recommender Systems*, pp. 139–166. Cham: Springer International Publishing, 2016.

[23] A. Rana and K. Deeba, "Online book recommendation system using collaborative filtering (with jaccard similarity)," *Journal of Physics: Conference Series*, vol. 1362, p. 012130, nov 2019.

[24] R. J. Ziarani and R. Ravanmehr, "Serendipity in recommender systems: A systematic literature review," *Journal of Computer Science and Technology*, vol. 36, pp. 375 – 396, 2021.

[25] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[26] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[27] Y. Xi, W. Liu, J. Lin, X. Cai, H. Zhu, J. Zhu, B. Chen, R. Tang, W. Zhang, R. Zhang, *et al.*, "Towards open-world recommendation with knowledge augmentation from large language models," *arXiv preprint arXiv:2306.10933*, 2023.

[28] J. Huang and K. Chen-Chuan Chang, "Towards reasoning in large language models: A survey," in *Findings of the Association for Computational Linguistics, ACL 2023*, Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 1049–1065, Association for Computational Linguistics (ACL), 2023. Publisher Copyright: © 2023 Association for Computational Linguistics.; 61st Annual Meeting of the Association for Computational Linguistics, ACL 2023 ; Conference date: 09-07-2023 Through 14-07-2023.

[29] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[30] J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, H. Zhang, Y. Liu, C. Wu, X. Li, C. Zhu, *et al.*, "How can recommender systems benefit from large language models: A survey," *arXiv preprint arXiv:2306.05817*, 2023.

[31] G. H. Torbati, A. Tigunova, A. Yates, and G. Weikum, "Recommendations by concise user profiles from review text," *ArXiv*, vol. abs/2311.01314, 2023.

[32] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017.

[33] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5)," *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022.

[34] H. Lyu, S. Jiang, H. Zeng, Y. Xia, and J. Luo, "Llm-rec: Personalized recommendation via prompting large language models," *ArXiv*, vol. abs/2307.15780, 2023.

[35] J. Lin, B. Chen, H. Wang, Y. Xi, Y. Qu, X. Dai, K. Zhang, R. Tang, Y. Yu, and W. Zhang, "Clickprompt: Ctr models are strong prompt generators for adapting language models to ctr prediction," *Proceedings of the ACM on Web Conference 2024*, 2023.

[36] J. Li, M. Wang, J. Li, J. Fu, X. Shen, J. Shang, and J. McAuley, "Text is all you need: Learning language representations for sequential recommendation," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, (New York, NY, USA), p. 1258â1267, Association for Computing Machinery, 2023.

[37] Z. Zhang and B. wei Wang, "Prompt learning for news recommendation," *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

[38] L. Wu, Z. Qiu, Z. Zheng, H. Zhu, and E. Chen, "Exploring large language model for graph data understanding in online job recommendations," *ArXiv*, vol. abs/2307.05722, 2023.

[39] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[40] X. Wang, K. Zhou, J.-R. Wen, and W. X. Zhao, "Towards unified conversational recommender systems via knowledge-enhanced prompt learning," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, (New York, NY, USA), p. 1929â1937, Association for Computing Machinery, 2022.

[41] L. Zou, S. Zhang, H. Cai, D. Ma, S. Cheng, S. Wang, D. Shi, Z. Cheng, and D. Yin, "Pre-trained language model based ranking in baidu search," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 4014–4022, 2021.

[42] A. Muhamed, I. Keivanloo, S. Perera, J. Mracek, Y. Xu, Q. Cui, S. Rajagopalan, B. Zeng, and T. M. Chilimbi, "Ctr-bert: Cost-effective knowledge distillation for billion-parameter teacher models," 2021.

[43] D. Wang, S. Yan, Y. Xia, K. Salamatian, W. Deng, and Q. Zhang, "Learning supplementary nlp features for ctr prediction in sponsored search," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, (New York, NY, USA), p. 4010â4020, Association for Computing Machinery, 2022.

[44] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, *et al.*, "Glm-130b: An open bilingual pre-trained model," *arXiv preprint arXiv:2210.02414*, 2022.

[45] J. E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *ArXiv*, vol. abs/2106.09685, 2021.

[46] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[47] S. Dai, N. Shao, H. Zhao, W. Yu, Z. Si, C. Xu, Z. Sun, X. Zhang, and J. Xu, "Uncovering chatgptâs capabilities in recommender systems," in *Proceedings of the 17th ACM Conference on Recommender Systems*, pp. 1126–1132, 2023.

[48] J. Lin, R. Shan, C. Zhu, K. Du, B. Chen, S. Quan, R. Tang, Y. Yu, and W. Zhang, "Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation," *Proceedings of the ACM on Web Conference 2024*, 2023.

[49] Y. Wang, Z. Jiang, Z. Chen, F. Yang, Y. Zhou, E. Cho, X. Fan, X. Huang, Y. Lu, and Y. Yang, "Recmind: Large language model powered agent for recommendation," *ArXiv*, vol. abs/2308.14296, 2023.

[50] H. Ding, Y. Ma, A. Deoras, B. Wang, and H. Wang, "Zero-shot recommender systems," *ArXiv*, vol. abs/2105.08318, 2021.

[51] Y. Hou, Z. He, J. McAuley, and W. X. Zhao, "Learning vector-quantized item representation for transferable sequential recommenders," *Proceedings of the ACM Web Conference 2023*, 2022.

[52] F. Liu, Y. Liu, Z. Cheng, L. Nie, and M. S. Kankanhalli, "Understanding before recommendation: Semantic aspect-aware review exploitation via large language models," *ArXiv*, vol. abs/2312.16275, 2023.

[53] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, and J. McAuley, "Bridging language and items for retrieval and recommendation," *ArXiv*, vol. abs/2403.03952, 2024.

[54] X. Wang, X. Tang, W. X. Zhao, J. Wang, and J. rong Wen, "Rethinking the evaluation for conversational recommendation in the era of large language models," *ArXiv*, vol. abs/2305.13112, 2023.

[55] Q. Liu, N. Chen, T. Sakai, and X.-M. Wu, "A first look at llm-powered generative news recommendation," *ArXiv*, vol. abs/2305.06566, 2023.

[56] J. Harte, W. Zorgdrager, P. Louridas, A. Katsifodimos, D. Jannach, and M. Fragkoulis, "Leveraging large language models for sequential recommendation," *Proceedings of the 17th ACM Conference on Recommender Systems*, 2023.

[57] S. Zhang, H. Yin, Q. Wang, T. Chen, H. Chen, and Q. V. H. Nguyen, "Inferring substitutable products with deep network embedding," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, p. 4306â4312, AAAI Press, 2019.

[58] J. Zheng, X. Wu, J. Niu, and A. Bolivar, "Substitutes or complements: another step forward in recommendations," in *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, (New York, NY, USA), p. 139â146, Association for Computing Machinery, 2009.

[59] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 1225â1234, Association for Computing Machinery, 2016.

[60] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, "Pme: Projected metric embedding on heterogeneous networks for link prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, (New York, NY, USA), p. 1177â1186, Association for Computing Machinery, 2018.

[61] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794, 2015.

[62] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, p. 993â1022, mar 2003.

[63] F. J. Ruiz, S. Athey, and D. M. Blei, "Shopper," *The Annals of Applied Statistics*, vol. 14, no. 1, pp. 1–27, 2020.

[64] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[65] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, *et al.*, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," *arXiv preprint arXiv:2403.05530*, 2024.

[66] Anthropic *et al.*, "The claude 3 model family: Opus, sonnet, haiku." https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, 2024. Online; accessed March 2024.

[67] T. Ma, Y. Cheng, H. Zhu, and H. Xiong, "Large language models are not stable recommender systems," *ArXiv*, vol. abs/2312.15746, 2023.

[68] H. Déjean, S. Clinchant, and T. Formal, "A thorough comparison of cross-encoders and llms for reranking splade," *arXiv preprint arXiv:2403.10407*, 2024.

[69] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

Appendix A

Further Details of Results

A.1 Experiment 2 Results

A.1.1 More Detailed Breakdown of Results on Different Cross-Encoder Inputs

Division	Week	Baselines				Cross-Encoders with Different Inputs							
		Naive Method		Bayesian Naive Method		title2title		title2feature		feature2feature		all2all	
		NDCG	$NDCG_f$	NDCG	$NDCG_f$	NDCG	$NDCG_f$	NDCG	$NDCG_f$	NDCG	$NDCG_f$	NDCG	$NDCG_f$
F	42	0.9283	0.5006	0.9312 (0.31)	0.4400 (12.09)	0.9295 (0.13)	0.6070 (21.26)	0.9295 (0.13)	0.6026 (20.39)	0.9292 (0.10)	0.5943 (18.72)	0.9293 (0.11)	0.5912 (18.12)
	44	0.9302	0.2746	0.9324 (0.24)	0.2256 (17.86)	0.9310 (0.09)	0.3431 (24.93)	0.9311 (0.09)	0.3385 (23.28)	0.9308 (0.06)	0.3340 (21.63)	0.9311 (0.10)	0.3246 (18.21)
	46	0.9326	0.5804	0.9353 (0.30)	0.5371 (7.46)	0.9339 (0.15)	0.6758 (16.43)	0.9337 (0.13)	0.6724 (15.84)	0.9338 (0.13)	0.6676 (15.01)	0.9338 (0.13)	0.6722 (15.81)
T	42	0.9284	0.3306	0.9283 (0.01)	0.3241 (1.95)	0.9292 (0.08)	0.3786 (14.52)	0.9297 (0.14)	0.3661 (10.73)	0.9294 (0.11)	0.3672 (11.06)	0.9293 (0.09)	0.3680 (11.31)
	44	0.9119	0.1371	0.9160(0.44)	0.1328 (3.14)	0.9121 (0.01)	0.1535 (11.99)	0.9122 (0.03)	0.1465 (6.86)	0.9120 (0.01)	0.1563 (13.99)	0.9123 (0.04)	0.1561 (13.86)
	46	0.9558	0.3493	0.9564 (0.07)	0.3511 (0.53)	0.9556 (0.01)	0.3966 (13.55)	0.9552 (0.06)	0.4011 (14.84)	0.9552 (0.07)	0.4057 (16.14)	0.9554 (0.04)	0.3953 (13.18)
B	42	0.9151	0.5641	0.9176 (0.28)	0.4756 (15.70)	0.9168 (0.18)	0.6705 (18.84)	0.9162 (0.12)	0.6674 (18.29)	0.9162 (0.12)	0.6743 (19.52)	0.9161 (0.10)	0.6693 (18.64)
	44	0.9262	0.3421	0.9268 (0.07)	0.2719 (20.51)	0.9266 (0.05)	0.4127 (20.64)	0.9266 (0.05)	0.4171 (21.92)	0.9269 (0.08)	0.4121 (20.48)	0.9269 (0.07)	0.4155 (21.48)
	46	0.9141	0.5757	0.9159 (0.19)	0.5116 (11.14)	0.9161 (0.21)	0.6678 (16.00)	0.9160 (0.20)	0.6705 (16.47)	0.9162 (0.22)	0.6718 (16.70)	0.9159 (0.19)	0.6712 (16.59)
W	42	0.8559	0.4159	0.8560 (0.01)	0.3660 (12.01)	0.8621(0.73)	0.5202 (25.06)	0.8610 (0.61)	0.5113 (22.92)	0.8612 (0.63)	0.5182 (24.59)	0.8606 (0.55)	0.5216 (25.42)
	44	0.8857	0.1317	0.8866 (0.10)	0.0993 (24.59)	0.8876 (0.21)	0.1701 (29.16)	0.8893 (0.40)	0.1693 (28.57)	0.8878 (0.23)	0.1729 (31.30)	0.8879 (0.24)	0.1745 (32.51)
	46	0.8922	0.2456	0.8931 (0.10)	0.2182 (11.14)	0.8940 (0.21)	0.3074 (25.20)	0.8938 (0.18)	0.3082 (25.51)	0.8945 (0.26)	0.3080 (25.44)	0.8941 (0.21)	0.3114 (26.79)
N	42	0.9095	0.4264	0.9108 (0.14)	0.3936 (7.69)	0.9113 (0.19)	0.5224 (22.51)	0.9114 (0.20)	0.5262 (23.40)	0.9112 (0.18)	0.5265 (23.48)	0.9110 (0.16)	0.5251 (23.14)
	44	0.9162	0.1976	0.9170 (0.09)	0.1756 (11.13)	0.9174 (0.13)	0.2485 (25.79)	0.9175 (0.15)	0.2509 (27.01)	0.9176 (0.15)	0.2541 (28.62)	0.9173 (0.13)	0.2518 (27.42)
	46	0.9134	0.5537	0.9145 (0.12)	0.5334 (3.66)	0.9161 (0.30)	0.6452 (16.52)	0.9162 (0.31)	0.6472 (16.88)	0.9162 (0.30)	0.6481 (17.05)	0.9158 (0.26)	0.6471 (16.86)

Table A.1: Results of Experiment 2 (Different Inputs to Cross-Encoders): The table displays the NDCG and $NDCG_f$ scores for cross-encoders with different inputs, including title2title, title2feature, feature2feature and all2all configurations over divisions F, T, B, W and N. Each score is followed by a percentage in parentheses, indicating the percentage increase (green) or decrease (red) relative to the naive baseline. The highest NDCG scores in each division for the week are highlighted in yellow, while the best $NDCG_f$ scores are highlighted in orange.

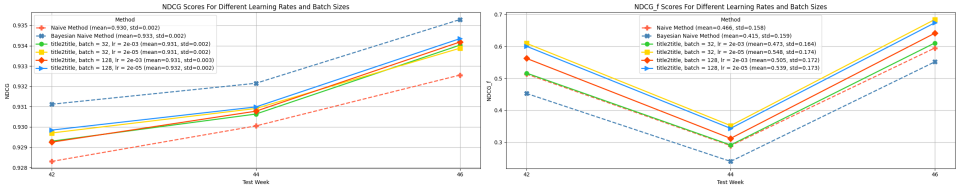
A.1.2 More Detailed Breakdown of Results on Finetuning Effect

Table A.2: Ablation Study Results on Finetuning Effect: Each entry presents (score of not fine-tuned model, score of fine-tuned model) and percentage changes between them. Positive changes are highlighted in green, and negative changes in red. The study spans divisions F, T, B, and W and test weeks 42, 44, and 46. The entry with the highest percentage increase in NDCG score is highlighted in yellow, and the highest in $NDCG_f$ score in orange. Note that numerical values are rounded to 3 significant figures.

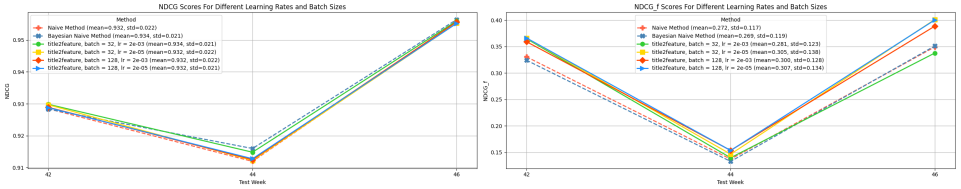
Division	Week	Cross-Encoders with Different Inputs							
		title2title		**title2feature**		**feature2feature**		**all2all**	
		NDCG	$NDCG_f$	NDCG	$NDCG_f$	NDCG	$NDCG_f$	NDCG	$NDCG_f$
F	42	(0.928, 0.930) +0.216%	(0.527, 0.592) +12.334%	(0.926, 0.927) +0.108%	(0.395, 0.439) +11.139%	(0.926, 0.927) +0.108%	(0.371, 0.397) +7.008%	(0.927, 0.929) +0.216%	(0.530, 0.603) +13.774%
	44	(0.930, 0.931) +0.108%	(0.278, 0.332) +19.424%	(0.928, 0.929) +0.108%	(0.199, 0.226) +13.568%	(0.928, 0.928) +0.0%	(0.183, 0.200) +9.290%	(0.929, 0.931) +0.215%	(0.281, 0.338) +20.285%
	46	(0.932, 0.934) +0.215%	(0.606, 0.664) +9.571%	(0.931, 0.932) +0.107%	(0.488, 0.524) +7.377%	(0.931, 0.932) +0.107%	(0.467, 0.487) +4.283%	(0.932, 0.934) +0.215%	(0.607, 0.671) +10.544%
T	42	(0.928, 0.929) +0.108%	(0.364, 0.379) +4.121%	(0.927, 0.930) +0.324%	(0.362, 0.366) +1.105%	(0.928, 0.929) +0.108%	(0.351, 0.367) +4.558%	(0.927, 0.929) +0.216%	(0.359, 0.368) +2.507%
	44	(0.910, 0.912) +0.220%	(0.146, 0.154) +5.479%	(0.913, 0.912) -0.110%	(0.149, 0.146) -2.013%	(0.908, 0.912) +0.441%	(0.136, 0.156) +14.706%	(0.911, 0.912) +0.110%	(0.145, 0.156) +7.586%
	46	(0.955, 0.956) +0.105%	(0.378, 0.397) +5.026%	(0.956, 0.955) -0.105%	(0.384, 0.401) +4.427%	(0.955, 0.955) +0.0%	(0.379, 0.406) +7.124%	(0.955, 0.955) +0.0%	(0.379, 0.395) +4.222%
B	42	(0.914, 0.917) +0.328%	(0.589, 0.670) +13.752%	(0.914, 0.916) +0.219%	(0.568, 0.667) +17.430%	(0.914, 0.916) +0.219%	(0.581, 0.674) +16.007%	(0.914, 0.916) +0.219%	(0.595, 0.669) +12.437%
	44	(0.926, 0.927) +0.108%	(0.337, 0.413) +22.552%	(0.925, 0.927) +0.216%	(0.322, 0.417) +29.503%	(0.926, 0.927) +0.108%	(0.339, 0.412) +21.534%	(0.926, 0.927) +0.108%	(0.345, 0.416) +20.580%
	46	(0.913, 0.916) +0.329%	(0.597, 0.668) +11.893%	(0.913, 0.916) +0.329%	(0.578, 0.671) +16.090%	(0.913, 0.916) +0.329%	(0.590, 0.672) +13.898%	(0.913, 0.916) +0.329%	(0.605, 0.671) +10.909%
W	42	(0.853, 0.862) +1.055%	(0.455, 0.520) +14.286%	(0.851, 0.861) +1.175%	(0.449, 0.511) +13.808%	(0.850, 0.861) +1.294%	(0.448, 0.518) +15.625%	(0.853, 0.861) +0.938%	(0.462, 0.522) +12.987%
	44	(0.886, 0.888) +0.226%	(0.140, 0.170) +21.429%	(0.885, 0.889) +0.452%	(0.136, 0.169) +24.265%	(0.883, 0.888) +0.566%	(0.140, 0.173) +23.571%	(0.884, 0.888) +0.452%	(0.142, 0.175) +23.239%
	46	(0.891, 0.894) +0.337%	(0.267, 0.307) +14.981%	(0.886, 0.894) +0.903%	(0.263, 0.308) +17.110%	(0.886, 0.894) +0.903%	(0.265, 0.308) +16.226%	(0.888, 0.894) +0.676%	(0.274, 0.311) +13.504%
N	42	(0.909, 0.911) +0.255%	(0.477, 0.522) +9.574%	(0.909, 0.911) +0.309%	(0.462, 0.526) +13.852%	(0.910, 0.911) +0.183%	(0.468, 0.527) +12.512%	(0.910, 0.911) +0.134%	(0.480, 0.525) +9.481%
	44	(0.916, 0.917) +0.166%	(0.216, 0.249) +15.2%	(0.916, 0.918) +0.171%	(0.206, 0.251) +22.066%	(0.916, 0.918) +0.125%	(0.209, 0.254) +21.773%	(0.916, 0.917) +0.089%	(0.216, 0.252) +16.649%
	46	(0.913, 0.916) +0.324%	(0.608, 0.645) +6.084%	(0.913, 0.916) +0.362%	(0.593, 0.647) +9.057%	(0.914, 0.916) +0.25%	(0.596, 0.648) +8.834%	(0.914, 0.916) +0.231%	(0.608, 0.647) +6.437%

A.2 Experiment 3 Results

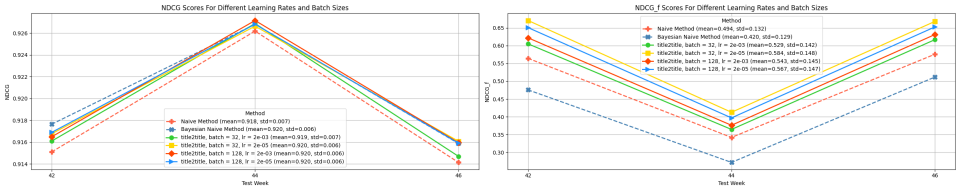
A.2.1 More Detailed Breakdown of Results on Hyperparamter Tuning



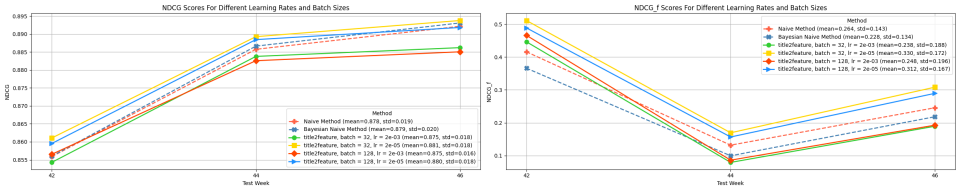
(a) Division F



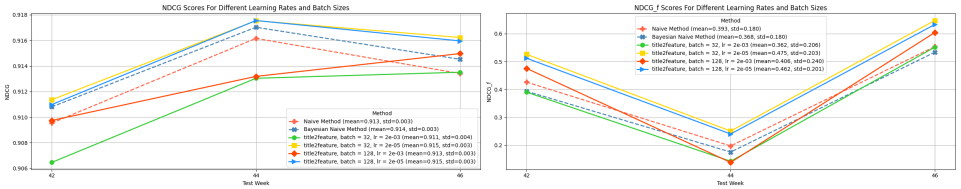
(b) Division T



(c) Division B



(d) Division W



(e) Division N

Figure A.1: Results for Hyper-parameter Tuning