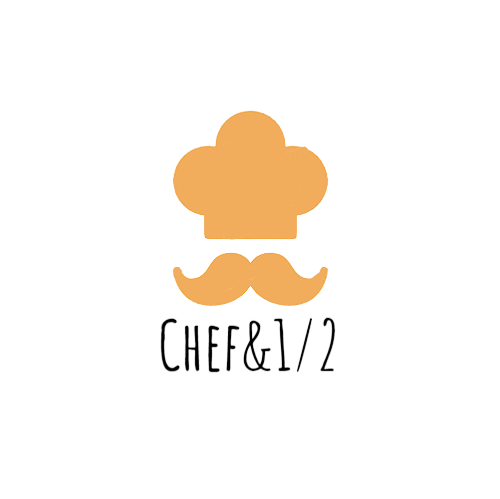


Cairo University

Faculty of Engineering

Department of Computer Engineering

**Chef w Nos**

A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

**Presented by**

Karim Amr Mohamed Amr

Abdallah Ahmed Omar Magdy

**Supervised by**

Dr.Magda Fayek

12-06-2023

All rights reserved. This report may not be reproduced in whole or in part, by photocopying or other means, without the permission of the authors/department.

Abstract

With the fast-paced lifestyle that we are living, eating has become an afterthought, people want to eat to get through the day and don’t want to spend time thinking and planning what to eat, which makes cooking even more difficult to do daily.

Figuring out what to eat every day is needless time wasting whether it’s just you eating alone or your whole family negotiating and discussing what to eat just to settle for an old recipe.

It leads to cooking the same recipes in a cycle which heavily limits recipe discovery and can turn cooking into more of a tedious chore.

After all the thinking and planning when you finally settle on a recipe it’s hard to follow the video’s steps and you end up pausing and rewinding which makes you wonder why I cooked in the first place.

To make the cooking process easier, faster, and more convenient the user should have fast access to recipes that he’s likely to make and that suit his inventory of ingredients, and be able to ask for certain recipes, ingredients, or cravings.

The user should also have a way to conveniently navigate the steps of the recipe that he chose to make.

We propose a system that accompanies the user and helps make cooking a simpler task.

The system should be able to chat with the user to understand their craving and can adjust on the fly to the user’s requests. The user’s request is then sent to a recommendation system that recommends recipes based on user’s request, favorite recipes, and past recipes, this process can go back and forth until the user is satisfied and chooses a recipe.

The recipe is then loaded and presented to the user as a collection of steps he can move through freely.

The system is presented in an application form where users can chat with their chef and can ask for similar recipes, recipes with certain ingredients and characteristics.

The application also provides relevant recommendations based on user’s likes and history.

An inventory system is also added to monitor ingredients and give users insights on their shopping and diet.

Recipe understanding is also presented to segment recipes video to logical steps to allow the user to freely navigate the recipe’s steps.

الملخص

مع نمط الحياة السريع الذي نعيشه، أصبح الأكل مجرد فكرة ثانوية، حيث يرغب الناس في تناول الطعام للمضي قدمًا في يومهم ولا يرغبون في قضاء الوقت في التفكير والتخطيط لما يجب تناوله. وهذا يجعل عملية الطبخ أكثر صعوبة في القيام بها يوميًا.

قرار ما يجب تناوله كل يوم يستهلك الكثير من الوقت عن طريق التفاوض ومناقشة العائلة بشأن ما يجب تناوله فقط للوصول إلى وصفة قديمة. هذا يؤدي إلى تكرار نفس الوصفات بشكل متكرر، مما يقيد اكتشاف وصفات جديدة ويجعل عملية الطبخ مهمة مملة.

بعد كل التفكير والتخطيط عندما تستقر على وصفة معينة، يكون من الصعب اتباع خطوات الفيديو وتجد نفسك توقف وإعادة المشاهدة، مما يجعلك تتساءل لماذا طبخت في المقام الأول.

لجعل عملية الطبخ أسهل وأسرع وأكثر ملائمة، يجب أن يتمكن المستخدم من الوصول السريع إلى وصفات يحتمل أن يقوم بتحضيرها وتتناسب مع مخزونه من المكونات، وأن يكون بإمكانه طلب وصفات معينة أو مكونات أو رغبات.

يجب أيضًا أن يكون للمستخدم وسيلة للتنقل بسهولة في خطوات الوصفة التي اختارها.

نقترح نظامًا يرافق المستخدم ويساعده على جعل الطبخ مهمة أبسط.

يجب أن يكون النظام قادرًا على محادثة المستخدم لفهم رغباته ويمكنه التكيف مع طلبات المستخدم. يتم إرسال طلب المستخدم إلى نظام توصية يوصي بوصفات استنادًا إلى طلب المستخدم والوصفات المفضلة والوصفات السابقة للمستخدم، ويمكن أن يتم هذا العمل ذهابًا وإيابًا حتى يكون المستخدم راضيًا ويختار وصفة.

ثم يتم تحميل الوصفة وتقديمها للمستخدم على هيئة مجموعة من الخطوات التي يمكنه التنقل فيها بحرية.

يتم تقديم النظام في شكل تطبيق حيث يمكن للمستخدمين التحدث مع طاهيهم وطلب وصفات مماثلة أو وصفات تحتوي على مكونات وخصائص معينة.

يوفر التطبيق أيضًا توصيات ذات صلة استنادًا إلى تفضيلات المستخدم وتاريخه.

يتم أيضًا إضافة نظام للجرد لمراقبة المكونات وتزويد المستخدمين بإرشادات حول التسوق والنظام الغذائي.

كما يتم تقديم فهم الوصفة لتقسيم فيديو الوصفة إلى خطوات منطقية للسماح للمستخدم بالتنقل بحرية في خطوات الوصفة.

ACKNOWLEDGMENT

First, we would like to thank God for helping us through this long and challenging journey. We would also like to thank our supervisors Dr. Magda Fayek for her support, guidance, and advice. During this journey we got a lot of support from our family and friends for which we are very thankful.

Table of Contents

[Abstract ii](#_Toc136796477)

[الملخص iii](#_Toc136796478)

[ACKNOWLEDGMENT iv](#_Toc136796479)

[Table of Contents v](#_Toc136796480)

[List of Figures ix](#_Toc136796481)

[List of Tables x](#_Toc136796482)

[List of Abbreviation xi](#_Toc136796483)

[List of Symbols xii](#_Toc136796484)

[Contacts xiii](#_Toc136796485)

[Chapter 1: Introduction 1](#_Toc136796486)

[1.1. Motivation and Justification 1](#_Toc136796487)

[1.2. The Essential Question 2](#_Toc136796488)

[1.3. Project Objectives and Problem Definition 2](#_Toc136796489)

[1.4. Project Outcomes 3](#_Toc136796490)

[1.5. Document Organization 3](#_Toc136796491)

[Chapter 2: Market Feasibility Study 4](#_Toc136796492)

[2.1. Targeted Customers 4](#_Toc136796493)

[2.2. Market Survey 4](#_Toc136796494)

[2.2.1. BASYL[3] 4](#_Toc136796495)

[2.2.2. Tasty[4] 6](#_Toc136796496)

[2.3. Business Case and Financial Analysis 7](#_Toc136796497)

[2.3.1. Competitive Analysis 7](#_Toc136796498)

[2.3.2. Business Case 7](#_Toc136796499)

[2.3.3. Financial Analysis 8](#_Toc136796500)

[Chapter 3: Literature Survey 9](#_Toc136796501)

[3.1. Recommender systems[5] 9](#_Toc136796502)

[3.1.1 Collaborative filtering[5] 10](#_Toc136796503)

[3.1.2 Content-based filtering [5] 11](#_Toc136796504)

[3.1.3 Context filtering[5] 12](#_Toc136796505)

[3.2. Natural language processing 13](#_Toc136796506)

[3.2.1 Recurrent neural network 13](#_Toc136796507)

[3.2.2 Gated Recurrent Unit (GRU) 14](#_Toc136796508)

[3.2.3 LSTM 15](#_Toc136796509)

[3.3. Transformers (Attention Is All You Need)[7] 16](#_Toc136796510)

[3.4. (BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding)[8] 18](#_Toc136796511)

[3.5. Personalized Embedding-based e-Commerce Recommendations at eBay[9] 19](#_Toc136796512)

[3.6. Dynamic Graph Neural Networks for Sequential Recommendation[10] 21](#_Toc136796513)

[3.7. Learning to Measure Changes: Fully Convolutional Siamese Networks for Scene Change Detection[11] 24](#_Toc136796514)

[3.8. Key-Frame Extraction Based on Histogram and Adaptive Clustering[13] 25](#_Toc136796515)

[3.9. Comparative Study of Previous Work 28](#_Toc136796516)

[3.10. Implemented Approach 30](#_Toc136796517)

[Chapter 4: System Design and Architecture 32](#_Toc136796518)

[4.1. Overview and Assumptions 32](#_Toc136796519)

[4.2. System Architecture 33](#_Toc136796520)

[4.2.1. Block Diagram 33](#_Toc136796521)

[4.3. Module 1: Chatbot 34](#_Toc136796522)

[4.3.1. Functional Description 34](#_Toc136796523)

[4.3.2. Modular Decomposition 35](#_Toc136796524)

[4.3.3. Design Constraints 38](#_Toc136796525)

[4.4. Module 2: Recommender system 39](#_Toc136796526)

[4.4.1. Functional Description 39](#_Toc136796527)

[4.4.2. Modular Decomposition 39](#_Toc136796528)

[4.4.3. Design Constraints 45](#_Toc136796529)

[4.5. Module 3: Video analysis 46](#_Toc136796530)

[4.5.1 Functional description 46](#_Toc136796531)

[4.5.2 Modular Decomposition 47](#_Toc136796532)

[4.5.3. Design Constraints 51](#_Toc136796533)

[Chapter 5: System Testing and Verification 53](#_Toc136796534)

[5.1. Testing Setup 53](#_Toc136796535)

[5.2. Testing Plan and Strategy 53](#_Toc136796536)

[5.2.1. Module Testing 54](#_Toc136796537)

[5.2.1.1 Recommender System 54](#_Toc136796538)

[5.2.1.2 Chatbot 56](#_Toc136796539)

[5.2.1.3 Video Analysis 57](#_Toc136796540)

[5.2.2. Integration Testing 58](#_Toc136796541)

[5.3. Testing Schedule 59](#_Toc136796542)

[5.4. Comparative Results to Previous Work 59](#_Toc136796543)

[Chapter 6: Conclusions and Future Work 62](#_Toc136796544)

[6.1. Faced Challenges 62](#_Toc136796545)

[6.1.1 Dataset 62](#_Toc136796546)

[6.1.1.1 Recipes dataset 62](#_Toc136796547)

[6.1.1.2 chatbot dataset 62](#_Toc136796548)

[6.1.2 Scene detection 63](#_Toc136796549)

[6.1.3 Bridging the Gap Between Desired Features and Implementation 63](#_Toc136796550)

[6.2. Gained Experience 63](#_Toc136796551)

[6.2.1 Enhancing search skills 63](#_Toc136796552)

[6.2.2 learned how to fine tune bert-model 63](#_Toc136796553)

[6.2.3 Teamwork and Planning 64](#_Toc136796554)

[6.3. Conclusions 64](#_Toc136796555)

[6.4. Future Work 64](#_Toc136796556)

List of Figures

[Figure ‎2‑1 BASYL Prompt 4](#_Toc136796569)

[Figure ‎2‑2 BASYL output 5](#_Toc136796570)

[Figure ‎2‑3 Tasty's recipe recommendations 6](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796571)

[Figure ‎2‑4 Tasty chat interface 6](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796572)

[Figure ‎3‑1 Recommender systems 9](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796573)

[Figure ‎3‑2 Collaborative filtering 10](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796574)

[Figure ‎3‑3 Content-based 11](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796575)

[Figure ‎3‑4 Contextual Filtering 12](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796576)

[Figure ‎3‑5 14](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796577)

[Figure ‎3‑6 14](#_Toc136796578)

[Figure ‎3‑7 15](#_Toc136796579)

[Figure ‎3‑8 16](#_Toc136796580)

[Figure ‎3‑9 18](#_Toc136796581)

[Figure ‎3‑10 Typical Two Tower model 20](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796582)

[Figure ‎3‑11 eBay recommendation model 20](#_Toc136796583)

[Figure ‎3‑12 Graph example 22](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796584)

[Figure ‎3‑13 DSGR Architecture 22](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796585)

[Figure ‎3‑14 Example Siamese network 24](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796586)

[Figure ‎3‑15 CosimeNet 25](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796587)

[Figure ‎3‑16 Example frame 26](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796588)

[Figure ‎3‑17 Frame as block 27](#_Toc136796589)

[Figure ‎3‑18 Centroids of clusters formed 27](#_Toc136796590)

[Figure ‎4‑1 Chef&1/2 high level architecture 33](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796591)

[Figure ‎4‑2 Chat interaction with the intent Ask 36](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796592)

[Figure ‎4‑3 Chat interaction with the intent Refuse 36](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796593)

[Figure ‎4‑4 NER example 37](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796594)

[Figure ‎4‑5 Chef&1/2 Chatbot 38](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796595)

[Figure ‎4‑6 Example of Vector Space[16] 42](#_Toc136796596)

[Figure ‎4‑7 Graph construction 44](#_Toc136796597)

[Figure ‎4‑8 Chef&1/2 Recommender system 45](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796598)

[Figure ‎4‑9 Color histograms for a given frame 47](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796599)

[Figure ‎4‑10 SVD Graphically 48](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796600)

[Figure ‎4‑11 Clusters formed 49](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796601)

[Figure ‎4‑12 Frames Clustered using KMeans 50](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796602)

[Figure ‎4‑13 Scene Change Detection Example 50](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796603)

[Figure ‎4‑14 Scene Change Detection Example 51](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796604)

[Figure ‎4‑15 Chef&1/2's Video analysis module 51](#_Toc136796605)

[Figure ‎5‑1 Example of k nearest recipes 54](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796606)

[Figure ‎5‑2 Video player showing first step 57](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796607)

[Figure ‎5‑3 Video player showing the next step 58](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796608)

[Figure ‎5‑4 Tasty vs Chef&1/2 Chatbot 60](file:///C:\Users\Fastora\Documents\GitHub\ChefwNos\GP%20Book%20Template%20-%20Credit.docx#_Toc136796609)

List of Tables

[Table ‎2‑1 Financial Analysis 8](#_Toc136796610)

[Table ‎3‑1 DSGR vs Two-Tower 28](#_Toc136796611)

[Table ‎3‑2 CosimeNet vs Keyframe extraction 29](#_Toc136796612)

[Table ‎4‑1 Intent example from own survey[14] 36](#_Toc136796613)

[Table ‎4‑2 NER model examples 37](#_Toc136796614)

[Table ‎4‑3 Elasticsearch results 40](#_Toc136796615)

[Table ‎4‑4 Comparison of Text to vector models 41](#_Toc136796616)

[Table ‎4‑5 Most similar to example 43](#_Toc136796617)

[Table ‎5‑1 DSGR evaluation 55](#_Toc136796618)

[Table ‎5‑2 Chatbot examples 56](#_Toc136796619)

[Table ‎5‑3 Testing Schedule 59](#_Toc136796620)

[Table ‎5‑4 DSGR' original vs own 60](#_Toc136796621)

List of Abbreviation

[The abbreviations should be put in an alphabetical order]

AI Artificial Intelligence

EA Evolutionary Algorithms

GA Genetic Algorithms

SA Simulated Annealing

VLSI Very Large Scale Integration

List of Symbols

Contacts

**Team Members**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **Phone Number** |
| Karim Amr Mohamed | Karimamr9@outlook.com | +2 01xxxxxxxxx |
| Abdallah Ahmed Hassan | [abc2@email.com](mailto:abc2@email.com) | +2 01xxxxxxxxx |
| Mohamed Amr Mohamed | [abc3@email.com](mailto:abc3@email.com) | +2 01xxxxxxxxx |
| Omar Magdy | [abc4@email.com](mailto:abc4@email.com) | +2 01xxxxxxxxx |

**Supervisor**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **Number** |
| Dr.Magda Fayek | [abc5@email.com](mailto:abc5@email.com) | +2 01xxxxxxxxx |

This page is left intentionally empty

# Introduction

The foods you choose to eat can have a direct impact on your ability to enjoy life to fullest. Perhaps the most obvious positive effect of food is the pleasurable feeling you get from eating a good-tasting meal, but to get there we have to go through a long process from first deciding what to cook until we sit down with our loved ones to eat what you have made.

Making food is no easy task it requires planning, knowledge and skill to cook something great we want to cut out the most boring parts of the experience so you can focus on the actual cooking and leave the rest out of your mind.

We all take part in cooking some way or another, whether we are the ones doing the actual cooking or being part of the discussion on what to eat today, a question that each one of us in exposed to daily and takes way more time than it is needed.

A smart cooking assistant would take care of these tedious details and would encourage people to try more recipes and discover whole new cooking styles, and would pull people back into cooking.

Having relevant recommendations in front of you and being able to ask for specific ingredients/characteristics and having all that within the items in your kitchen would greatly improve the cooking process.

We set out to simplify the planning and knowledge needed to start cooking by providing the users with tailored recipe recommendations based on their history, preferences and specific requests, while making following a recipe’s video more manageable while cooking.

The system proposed can interact with the users to get their specific requests if any using a chatbot that answers that understand requests and queries a recommendation system.

After choosing the right recipe the recipe’s video is analyzed and divided to steps to allow for smoother navigation.

## Motivation and Justification

Cooking is a not an easy task by any means due to the amount of planning, knowledge and skills needed to pull off a successful recipe and we are living in a fast-paced environment where food, groceries and taxies are being ordered by a push of a button, modernizing the cooking process would take out the boring parts leaving you with the pleasure of cooking a dish and the satisfaction that follows.

During late 2020 and early 2021 one of the common trends across the world was cooking at home[1] and that coincides with COVID-19 pandemic, people were at home and used this as an opportunity to level up their cooking skills.

Moreover, people between the age of 25-34 years old cook with either their smartphones or tablets[2] and the smartphone is becoming the ultimate sous-chef for millennials as they prefer to experience the whole culinary process from start to finish.

So, it’s apparent that people are already on their phones to look for the next recipes to make and the proposed application would fit perfectly into this category of helping people focus on cooking and exploring.

## The Essential Question

We all end up eating at the end of the day whether by cooking or ordering food but the question that we repeat to ourselves every day is what to eat today?

And eventually the main problem we are tackling, we want to eliminate this question and keep it to a minimum and we believe as engineers we should always look to make people’s life easier and more convenient by automating tedious tasks using technology.

And this mission perfectly aligns with the problem at hand it takes such an important aspect of our lives and simplifies it and make it more modern.

## Project Objectives and Problem Definition

We set out to simplify the planning and knowledge needed to start cooking so our users can focus on the cooking process itself.

The system should take care of the decision-making process as much as possible so the user can quickly start making the recipe, it also needs to react to feedback from user and receive specific requests to be as accurate as possible.

Keeping in mind the changing taste of the users the system should be able to adapt overtime to the users’ taste.

It’s important also that the recipes presented to the user are feasible, meaning that the ingredients are available in the kitchen.

In line with simplifying the knowledge needed to cook we’ll use videos that show step by step the recipe and the user can navigate easily through the video whilst following it.

Chef W Nos’s main focus is to help you decide your recipe for the day, we assume that the chatbot will only be used to ask for recipes and get positive or negative feedback not for general chatting or explaining steps of the recipe.

## Project Outcomes

The outcome of this project is a software application that accompanies the user on their android phone with a simple graphical user interface.

The user can see right away his recommended recipes that he can browse through, view their steps and attached videos.

If the user has a specific craving, he can then use the chatbot to ask for recommendation based on that craving, after choosing the recipe he’ll be directed to the recipe’s steps and segmented video that he can then navigate through.

## Document Organization

In Chapter 1 of this document, we provided an introduction to the project, discussing our motivation and reasons for undertaking it. We also introduced the key questions that our project aims to answer and outlined the project objectives and problem definition. Lastly, we discussed the expected outcomes of the project and identified the beneficiaries who will derive value from it.

Chapter 2 will focus on conducting a market study, where we will identify our target customers, analyze the competition in the market, and develop a comprehensive business plan and business case.

Chapter 3 will present a literature survey, providing both technical and non-technical background information necessary to understand the methods employed in our project. We will also discuss the relevant research papers that have influenced our implementation approach.

In Chapter 4, we will delve into the system design and architecture, starting from a high-level overview down to the detailed description of each independent module. We will explain how the system is designed to meet user requirements and provide an optimal user experience.

Chapter 5 will be dedicated to explaining the testing and verification steps employed to ensure the accuracy and correctness of the system outcomes. We will describe the testing procedures and methodologies used to validate the results.

Finally, in Chapter 6, we will conclude the document by summarizing the project, highlighting its features and limitations. We will also discuss potential avenues for further improvement and enhancement of the project.

# Market Feasibility Study

In this chapter we’ll discuss the market feasibility study, going into details into the targeted customers, market survey and talk about the business case and financial analysis.

Cooking assistants are pretty common in the market, when we first started to develop this idea, we looked at what other applications offered, gaining valuable insights that helped us come up with the set of features present in our applications.

## 2.1. Targeted Customers

As we discussed in the last chapter more people now are using their smartphones to help them in cooking, research from McGarry Bowen and Kraft Foods, the found that 59% of 25–34-year-olds cook with their smartphones[2].

So young people that are already using their smartphones for everything are one of our targeted customers.

Another group that we are very interested in is household parents, families tend to have the most discussion about what to eat today and with varying tastes and ingredients they would be benefit greatly from the application’s features.

## 2.2. Market Survey

In this section we’ll explore similar applications that provide cooking assistance to the user.

### 2.2.1. BASYL[3]

Basyl is an AI cooking assistant that generates recipes based on ingredients entered and can save generated recipes in a cookbook.

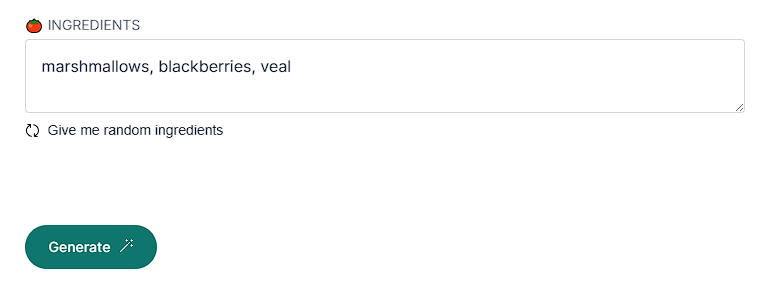


Figure ‑ BASYL Prompt

****

Figure ‑ BASYL output

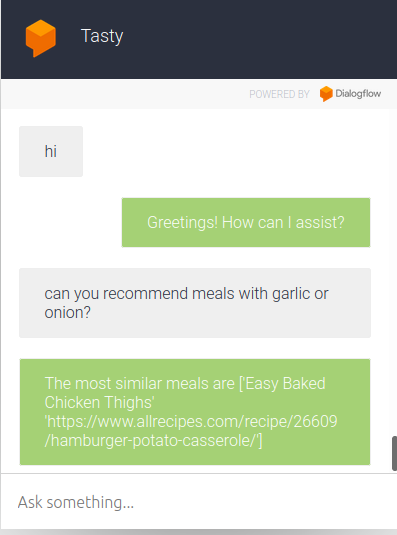
**Features:**

* Generate delicious custom recipes based on your preferences.
* Save your favorite recipes to your personalized cookbook.

**Drawbacks:**

* No recipe recommendations.
* No inventory system.

### 2.2.2. Tasty[4]

A project about recommending recipes and ingredients based on the needs of the user.

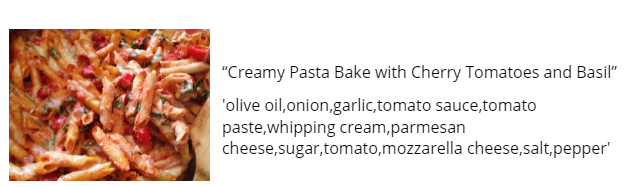
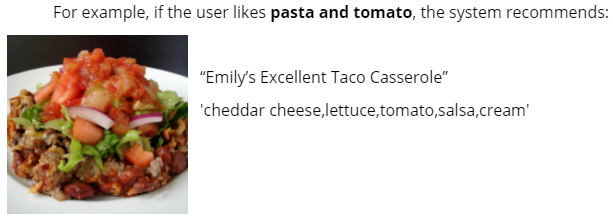


Figure ‑ Tasty's recipe recommendations

Figure ‑ Tasty chat interface

**Features:**

* Chatbot to interact with user.
* Get recipe recommendations with specified ingredients.
* Get similar ingredients and substitutions.

**Drawbacks:**

* No inventory system
* Doesn’t keep history of recipes.
* Redirects to recipe in text form.

## 2.3. Business Case and Financial Analysis

### 2.3.1. Competitive Analysis

Based on our market research, we have identified that there is no direct competitor that offers all the features present in our application, particularly considering our application's main language being Arabic. While we found several applications that offer some of the features included in our application, none of them provide the complete set of features we offer. Additionally, there are certain features unique to our application that we did not find in any existing applications.

The closest competitor to our application is "Tasty." Like our application, Tasty also includes a recommendation system, step-by-step video analysis, and a "what's in your kitchen" feature. However, there are notable differences between the two. One major distinction is the absence of a chatbot in Tasty, which is a key component of our application. The chatbot allows users to interact and request specific recipes, providing a more personalized and interactive experience. Furthermore, there are variations in the recommendation systems employed by the two applications, indicating differences in the underlying algorithms and approaches.

Overall, while there may be some applications that offer overlapping features with our application, none of them encompass the full range of capabilities we provide. Our application aims to combine these features seamlessly, catering to the specific needs of Arabic-speaking users and offering a comprehensive and unique cooking experience. By leveraging these differentiating factors, we have the opportunity to capture a significant market share and establish ourselves as a leading player in the culinary application domain.

### 2.3.2. Business Case

The market research has revealed an exciting opportunity to develop an AI cooking assistant that caters to the latest trends in home cooking. With more people eager to start cooking and experimenting in the kitchen, leveraging smartphones as a tool for assistance becomes crucial.

The application's main selling point is its accessibility, and to ensure widespread adoption, it will be offered completely free of charge. The revenue model will be based on partnerships with hypermarkets and online markets, where users can conveniently do their grocery shopping within the app. Leveraging the valuable information from the user's meal history and most frequently used ingredients, the application can provide personalized recommendations for grocery purchases, making the cooking process even more streamlined.

Looking ahead, the project envisions a future where the application goes beyond a mere assistant and becomes a comprehensive meal planning and preparation solution. This future service could offer the convenience of having suggested meals delivered directly to the user's home, eliminating the need for planning and preparation. This subscription-based model would provide users with a fixed number of meals per month that align with their dietary preferences and requirements, allowing them to focus solely on the joy of cooking.

By continuously exploring intelligent ways to improve the cooking process both within the application and beyond, the project aims to revolutionize the way people approach cooking and make it a more enjoyable and convenient experience.

### 2.3.3. Financial Analysis

Table ‑ Financial Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Measure | Year 1 | Year 2 | Year 3 | Year 4 |
| Application host & cloud server | $500 | $1,000 | $1,500 | $2,000 |
| Salaries | $6,000 | $8,000 | $10,000 | $12,000 |
| Office rent & bills | $2,500 | $2,500 | $4,300 | $4,300 |
| Advertising | $250 | $250 | $250 | $250 |
|  |  |  |  |  |
| Total cash out | $9,250 | $11,750 | $16,050 | $18,550 |
|  |  |  |  |  |
| Subscriptions profit | $1,000 | $3,000 | $8,000 | $13,000 |
| Application ads profit | $1,000 | $2,000 | $4,000 | $6,000 |
|  |  |  |  |  |
| Total cash in | $2,000 | $5,000 | $12,000 | $19,000 |
|  |  |  |  |  |
| Revenue | -$7,250 | -$6,750 | -$4,050 | $450 |

# Literature Survey

In this chapter we will introduce the topics needed to understand the project and we’ll also discuss the various papers, studies and methods explored while developing our system and speak about the challenges and difficulties faced along the way.

Going through these topics we’ll discover the importance of machine learning and AI in advancing multiple fields like computer vision, natural language processing and recommendations systems.

## 3.1. Recommender systems[5]

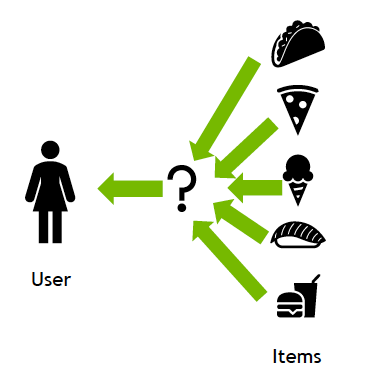
A recommendation system (or recommender system) is a class of machine learning that uses data to help predict, narrow down, and find what people are looking for among an exponentially growing number of options.[6]

Figure ‑ Recommender systems

The recommendations produced can be based on various criteria, including past purchases, search history, demographic information, and other factors.

Recommender systems have gained popularity among content and product providers due to their ability to analyze user interactions, such as impressions, clicks, likes, and purchases, in order to understand individual preferences and characteristics. By leveraging this data, these systems can make highly personalized predictions about consumer interests and desires. As a result, they can effectively steer consumers towards a wide range of products and services, including books, videos, health classes, and clothing.

### 3.1.1 Collaborative filtering[5]

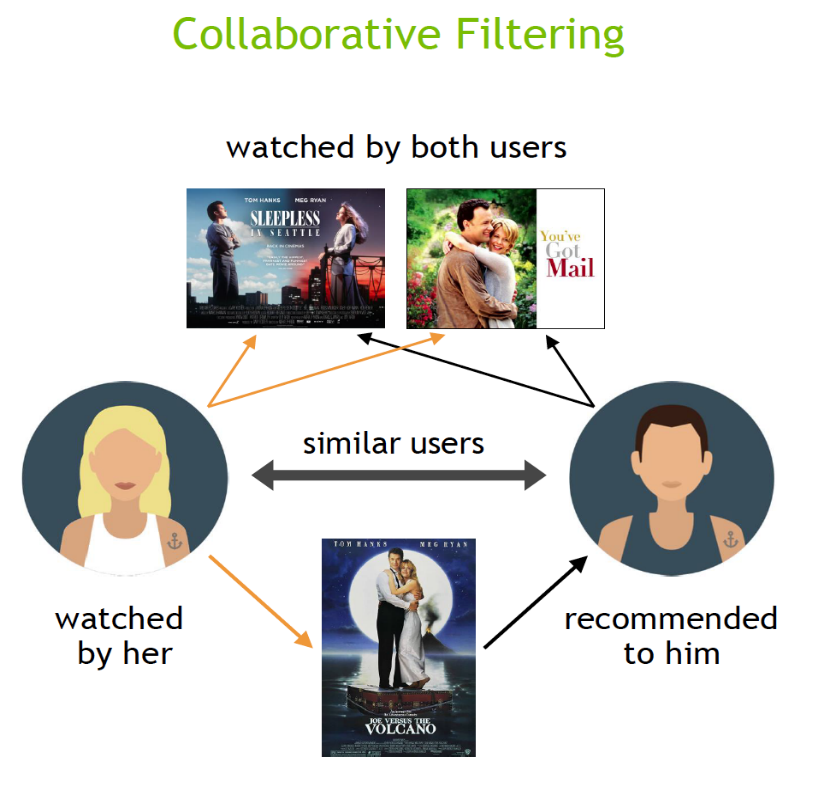
The collaborative filtering approach in recommender systems utilizes preference information from multiple users to recommend items. By analyzing the similarity of user preference behavior and considering previous interactions between users and items, these algorithms learn to predict future interactions. These systems create a model based on a user's past behavior, such as their previous purchases or ratings given to items, as well as similar decisions made by other users. The underlying concept is that if multiple users have made similar decisions and purchases in the past, such as choosing the same movie, there is a high likelihood that they will agree on future selections. For instance, if a collaborative filtering recommender system recognizes that you and another user have similar tastes in movies, it may suggest a movie to you that it knows the other user already enjoys.

Figure ‑ Collaborative filtering

### 3.1.2 Content-based filtering [5]

The content filtering approach in recommender systems relies on the attributes or features of an item to recommend other items that are similar to the user's preferences. This approach is based on analyzing the similarity between the attributes of items and the user's preferences. By considering information about a user and the items they have interacted with, such as a user's age, the cuisine category of a restaurant, or the average review for a movie, the recommender system models the likelihood of a new interaction.

For instance, if a content filtering recommender system observes that you enjoyed movies like "You've Got Mail" and "Sleepless in Seattle," it may recommend another movie to you with similar genres and/or cast, such as "Joe Versus the Volcano." The system takes into account the shared attributes or features of these movies, such as their genres or actors, to suggest other items that align with your preferences.

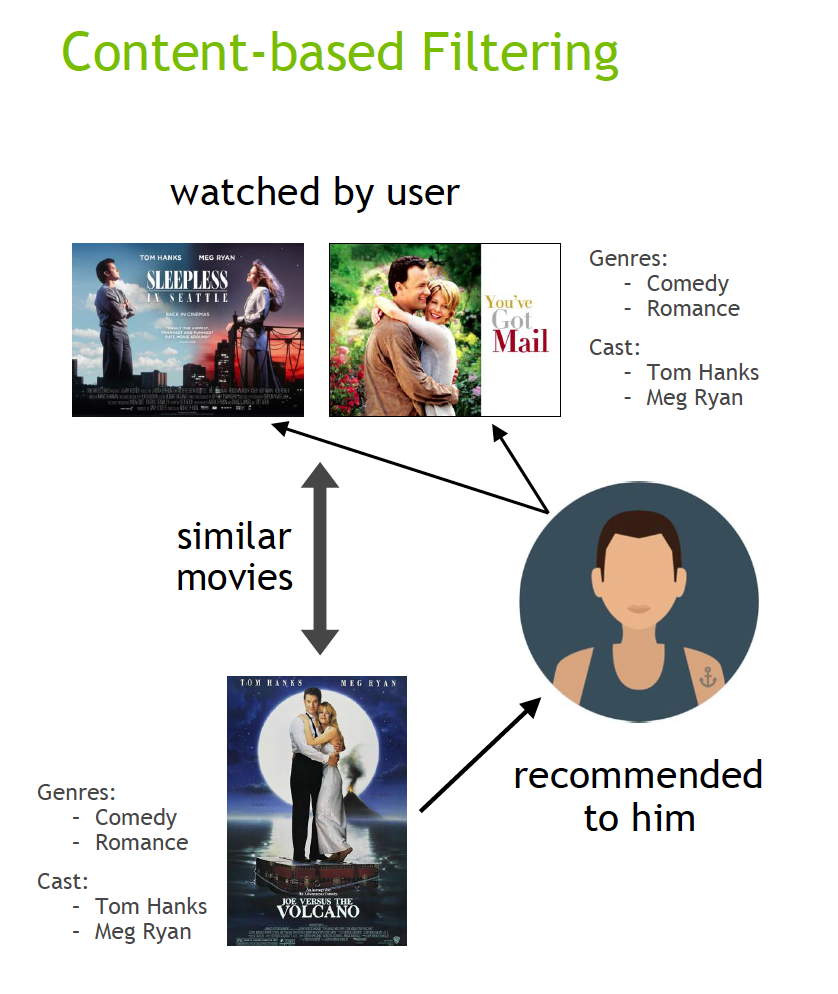


Figure ‑ Content-based

### 3.1.3 Context filtering[5]

In order to improve recommendations, Netflix has explored incorporating users' contextual information into the recommendation process. During a presentation at NVIDIA GTC, Netflix discussed their approach of framing recommendations as contextual sequence predictions. This method utilizes a sequence of contextual user actions, along with the current context, to estimate the likelihood of the next action.

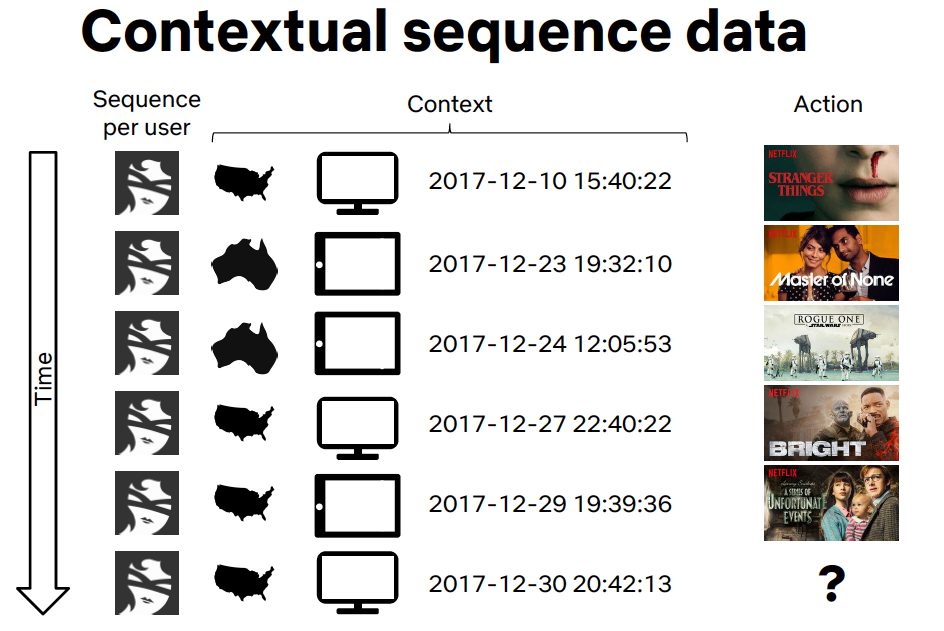
For instance, in the case of Netflix, they considered a sequence of contextual information for each user, including their country, device, date, and time when they watched a movie. By training a model using this sequence, Netflix aimed to predict what the user is likely to watch next. This approach takes into account the specific context of each user, allowing for more personalized and accurate recommendations based on their past viewing behaviors and contextual factors.

Figure ‑ Contextual Filtering

## 3.2. Natural language processing

Historically, computers have faced challenges in truly understanding human language. While they can collect, store, and process text inputs, they often lack the fundamental context and comprehension required for effective language understanding.

To address this limitation, the field of Natural Language Processing (NLP) emerged. NLP is a branch of artificial intelligence that focuses on equipping computers with the ability to read, analyze, interpret, and derive meaning from text and spoken words. It encompasses various techniques and methodologies that combine elements of linguistics, statistics, and Machine Learning to enable computers to better understand and work with human language.

By leveraging NLP, computers can go beyond basic text processing and start to understand the intricacies of language, such as syntax, semantics, and context. This allows for a range of applications, including language translation, sentiment analysis, question-answering systems, chatbots, and more. NLP plays a crucial role in enabling computers to "understand" and interact with human language more effectively.

NLP went through many advances that enabled greater efficiency and accuracy due to the introduction of new machine learning techniques, so we are only concerned with the latest advancements.

### 3.2.1 Recurrent neural network

Recurrent neural networks (RNN) differ from regular neural network as the input to a RNN is a single word instead of a whole sentence, meaning the network can handle varying lengths of sentences.

RNNs also share feature learned across different positions of text it does so by treating each word of a sentence as a sperate input that happens at time , the input at time and the activation value at .

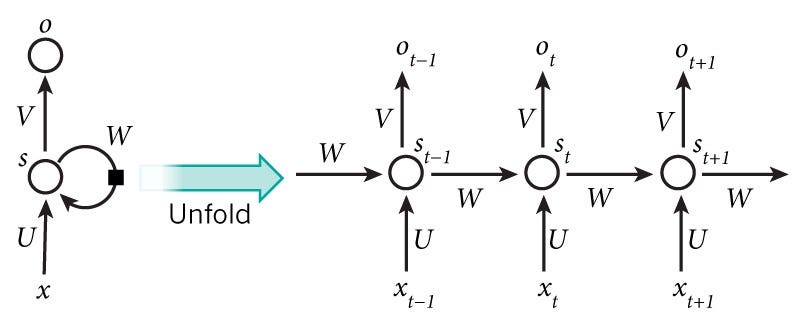


Figure ‑

### 3.2.2 Gated Recurrent Unit (GRU)

GRU (Gated Recurrent Unit) is a modified version of the basic recurrent unit used in neural networks. It addresses two important challenges: capturing long-range dependencies and overcoming the vanishing gradient problem.

The GRU introduces an update gate and an additional unit with tanh activation to improve its ability to capture long-range dependencies. This gate, along with the tanh unit, facilitates effective memory cell updating and helps overcome the vanishing gradient problem, leading to more robust and efficient training of the neural network.

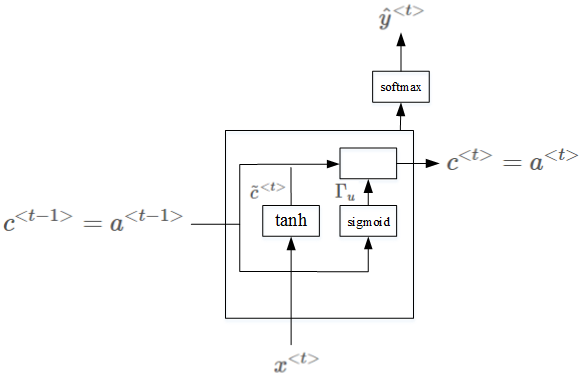


Figure ‑

### 3.2.3 LSTM

LSTM (Long Short-Term Memory) is a type of recurrent neural network architecture that is capable of addressing the challenges of capturing long-range dependencies and solving the vanishing gradient problem. It achieves this by using two gates: an update gate and a forget gate. The update gate knows how much of the new information should be incorporated into the memory cell, while the forget gate controls the extent to which the old information should be forgotten. This flexibility allows LSTM to effectively retain relevant information and discard irrelevant or outdated information, making it suitable for capturing long-term dependencies in sequential data.

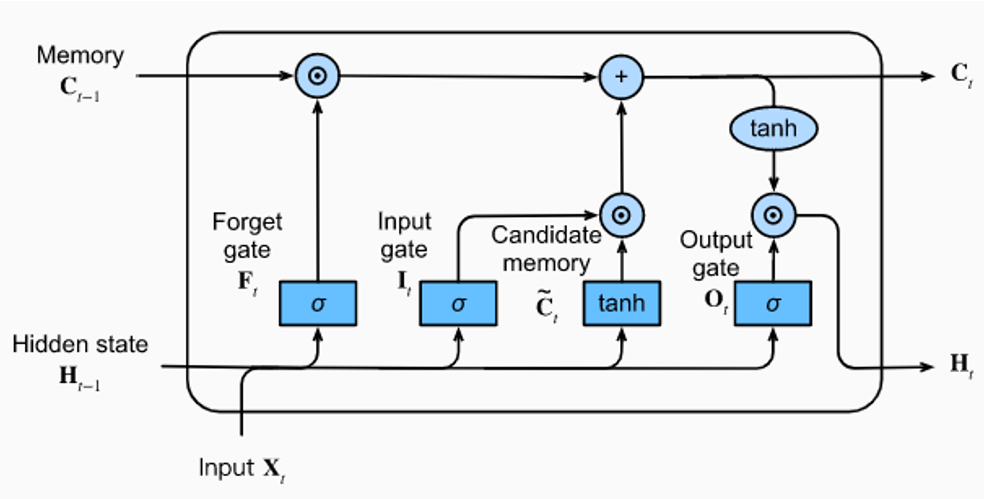


Figure ‑

## 3.3. Transformers (Attention Is All You Need)[7]

Figure ‑

Transformers is a type of neural network architecture that revolutionized natural language processing tasks. The concept of transformers was introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017.

Traditional sequence-based models, such as recurrent neural networks, suffer from limitations like sequential computation and some obstacles in capturing long-range dependencies. Transformers focused on addressing these issues by making a mechanism called self-attention.

In a transformer, the input sequence is first embedded into vectors known as embeddings. These embeddings are then passed through multiple layers of self-attention and feed-forward neural networks which they are processed in it. Self-attention allows the model to weigh the importance of different words in a sequence when generating representations, enabling it to focus on the most relevant words for each task.

The self-attention mechanism calculates attention scores between all pairs of words in a sequence. It determines how much each word is related to other words in the sequence during processing. The attention scores are computed based on the similarity of the embeddings and can be thought of as a measure of the importance of one word to another. After the self-attention step, the transformer applies feed-forward neural networks to each word's representation individually. These networks learn to transform and refine the representations. This process is repeated multiple times through stacked layers, allowing the model to figure the complex patterns and dependencies in the data.

One crucial aspect of transformers is the idea of positional encoding. Since transformers lack the sequential nature of recurrent neural networks, positional encoding is introduced to provide information about the order of words in the input sequence. This allows the model to differentiate between words based on their relative positions.

The "Attention Is All You Need" paper introduced the transformer architecture and demonstrated its effectiveness on various natural language processing tasks, such as machine translation. It showcased the superiority of transformers in capturing long-range dependencies, scalability to larger datasets, and parallelization capabilities compared to traditional recurrent neural networks.

Overall, transformers have become a fundamental architecture in the field of natural language processing, enabling breakthroughs in tasks such as machine translation, sentiment analysis, text summarization, and more. The self-attention mechanism and stacked layers of feed-forward networks in transformers have proven to be highly effective in capturing complex patterns and dependencies in sequential data.

## 3.4. (BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding)[8]

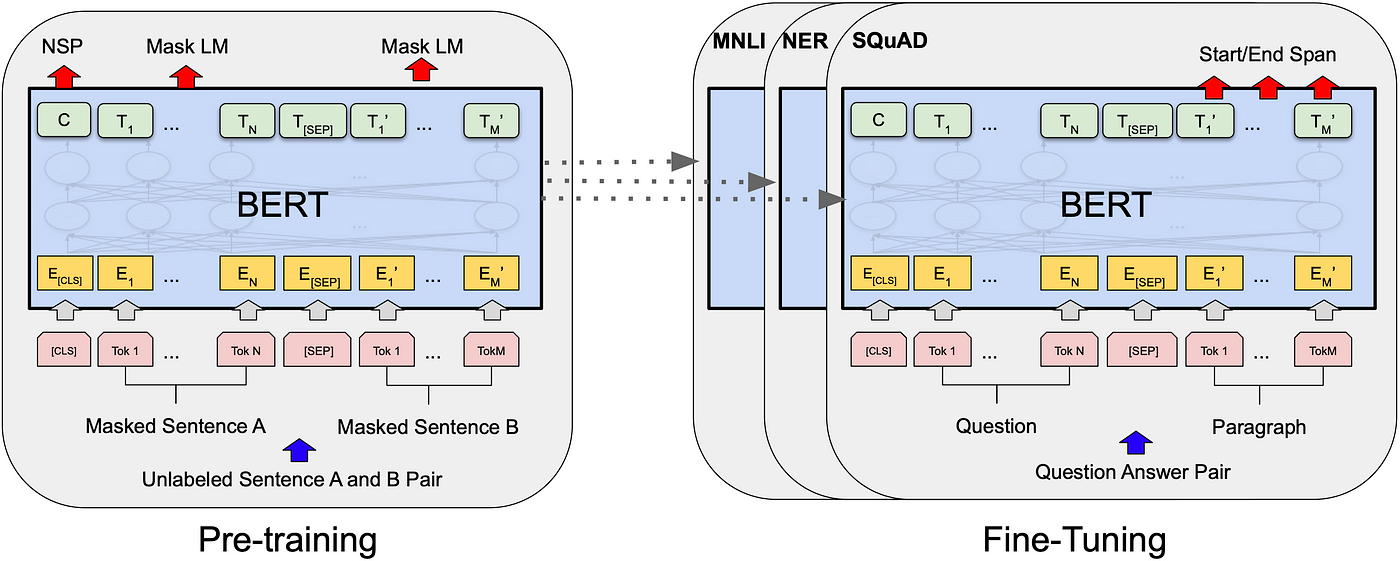


Figure ‑

BERT (Bidirectional Encoder Representations from Transformers) is a popular language model introduced in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. in 2018. It is a groundbreaking model that has significantly advanced natural language processing tasks.

Unlike traditional language models that process text in a sequential manner, BERT utilizes a transformer architecture and employs a pre-training and fine-tuning approach. BERT is "bidirectional" because it considers both the left and right contexts of each word during pre-training.

During pre-training, BERT is trained on large amounts of unlabeled text from various sources, such as books and websites. It learns to predict missing words in sentences by using the surrounding context. This process helps BERT develop a deep understanding of language and its nuances.

BERT's architecture consists of multiple transformer layers. Each layer includes self-attention mechanisms and feed-forward neural networks. The self-attention mechanism allows BERT to weigh the importance of different words in a sentence when generating contextualized word representations.

After pre-training, BERT is fine-tuned on specific downstream tasks, such as text classification, named entity recognition, question answering, and sentiment analysis. During fine-tuning, BERT is trained on labeled task-specific data, adjusting its parameters to make predictions for the target task.

One of the key features of BERT is its ability to capture contextual information effectively. It learns contextualized word representations that take into account the surrounding words in a sentence, which helps in understanding complex sentence structures and resolving word ambiguities.

BERT also introduces the concept of "masked language modeling" during pre-training. It randomly masks out some of the words in the input sentence and trains the model to predict those masked words based on the remaining context. This technique helps BERT to handle bidirectionality and learn deeper contextual relationships.

The impact of BERT has been significant in natural language processing, as it has achieved state-of-the-art results on various benchmark datasets. BERT's ability to capture contextual information and its transfer learning capabilities have made it a versatile and widely adopted model for various NLP tasks.

In summary, BERT is a language model that utilizes transformer architecture, pre-training, and fine-tuning to achieve remarkable results on natural language processing tasks. It learns contextualized word representations by considering bidirectional context and leverages large-scale pre-training data to develop a deep understanding of language.

## 3.5. Personalized Embedding-based e-Commerce Recommendations at eBay[9]

Items recommendations is an essential component of e-commerce marketplaces and given the scale and extreme sparsity of user-item matrix, meaning large number of new items being added and users don’t get chance to review them, the traditional collaborative filtering methods produce poor results.

Consequently, implicit user feedback such as clicks and purchases are also sparse and the traditional methods don’t fully capitalize on their significance.

In this paper an approach for generating personalized item recommendations is offered, by leaning to embed items and users in the same vector space to help combat the huge sparsity in the user-item matrix.

Item and user embeddings are computed using content features and multi-modal onsite user activity.

The model proposed for personalized recommendations is based on training a two-tower deep leaning model to generate user and item embeddings at the same time.

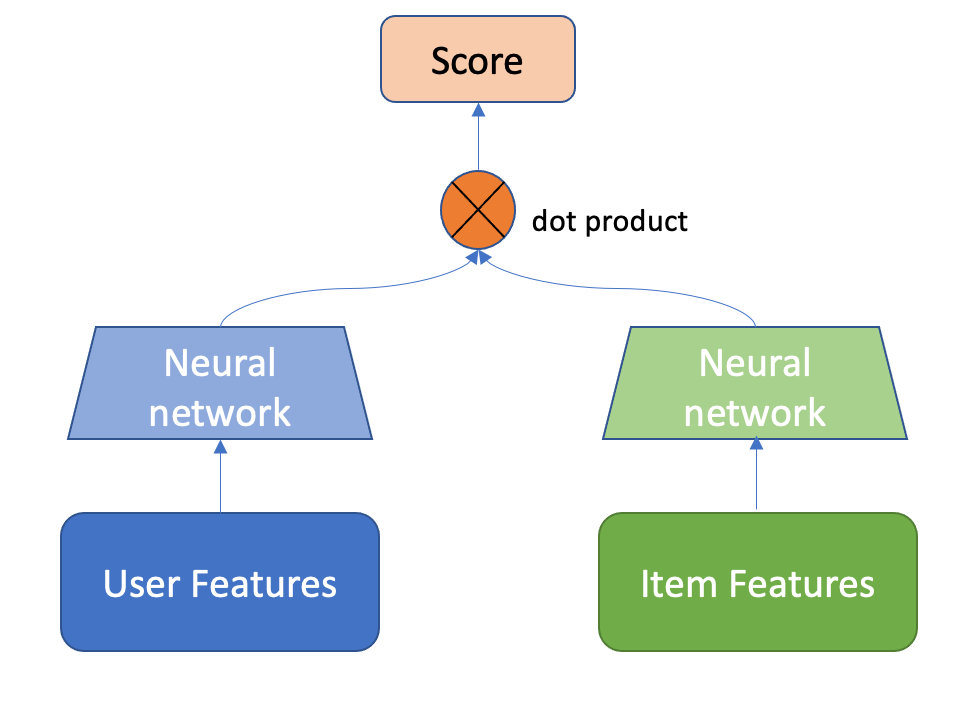


Figure ‑ Typical Two Tower model

The two-tower model is widely used for generating personal recommendations, it uses information for both the users and items together, the output of each neural network is a vector that represents user and item respectively then dot product is calculated and used as the rating of the item given by the user.

A loss function is then used to minimize the error between the actual rating and the calculated rating.

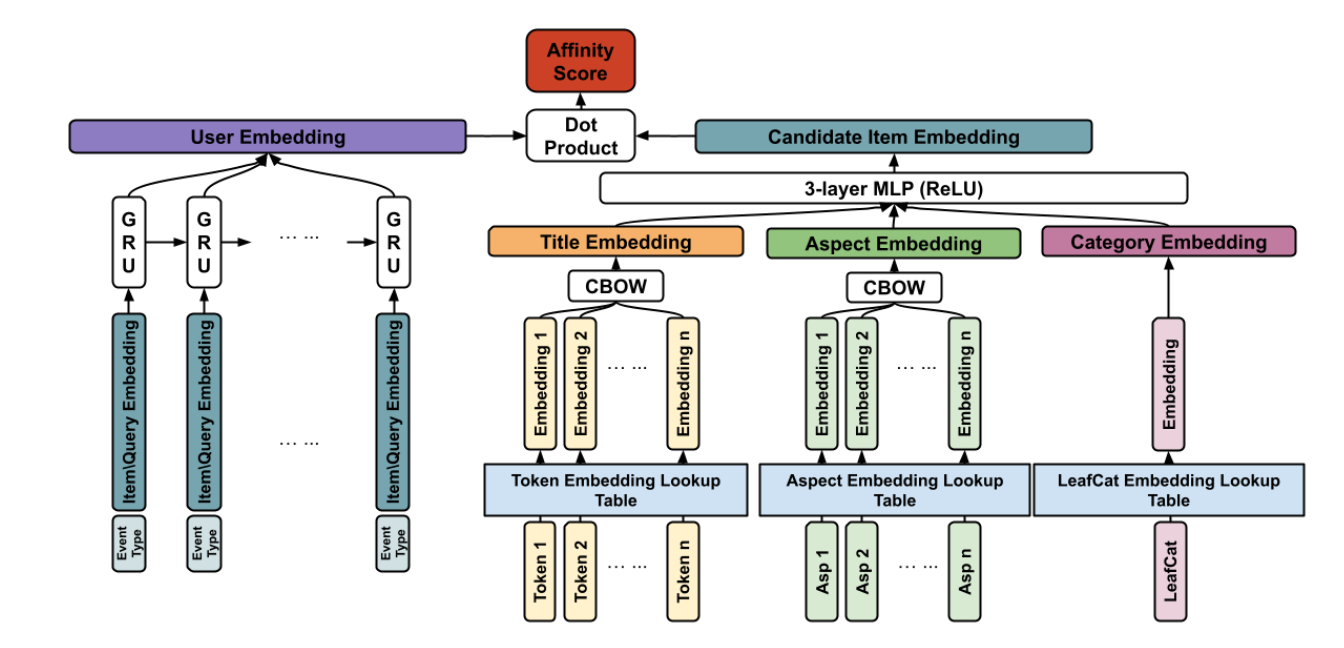


Figure ‑ eBay recommendation model

First, we’ll discuss the items network, an item at eBay corresponds to a listing, the item is represented as its content-based features such as its title, category and aspects.

Since title and aspects are text data, they are tokenized then converted to feature vectors using Continuous-Bag-Of-Words approach.

A CBOW vector is produced for each item feature then all vectors are concatenated and passed through a 3 layers neural network with ReLU activation to produce the item final embedding.

Secondly the user network, eBay considers some actions as valuable signals for the generation of recommendations such as viewing items, making a search query, adding an item to their shopping cart or their Wishlist, these actions are represented using a recurrent neural network that has access to the sequence of the events.

Each event has a GRU cell that produces output according to the following:

Where is the user event and its vector representation in .

The last part of the model is the Affinity function 𝛾 (v𝑖, u), the affinity is calculated between user U and item and it is constructed by the dot product between the user and item embeddings.

Both embeddings are normalized to have unit length and their dot product is constrained to be between -1 and 1 to distinguish between positive and negative items.

During the prediction stage, given the user embedding and a pool of candidate item embeddings the model would then sort the items based on user preference resulting in a more personal experience.

## 3.6. Dynamic Graph Neural Networks for Sequential Recommendation[10]

Modeling user preference from historical sequences is one of the core problems of sequential recommendation. Existing methods in this field are widely distributed from conventional methods to deep learning methods. However, most of them only model users’ interests within their own sequences and ignore the dynamic collaborative signals among different user sequences, making it insufficient to explore users’ preferences.

The paper proposes a new method named DSGR which connects different user sequences through a dynamic graph structure exploring the interactive behavior of users and items with time and order information.

DSGR leverages the collaborative information among different user sequences not only each user individually.

For example, in the figure below at time user1 interacts with , and while also having high order connection with user2 and user3 we can use this connection in predicting user1’s sequence.

DSGR also considers the dynamic influence of the high-order collaboration information at different times, user1’s graph changes overtime.

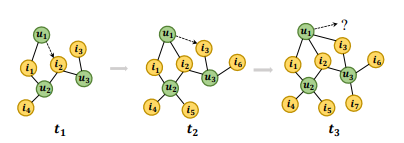


Figure ‑ Graph example

The DSGR model’s architecture has 4 components:

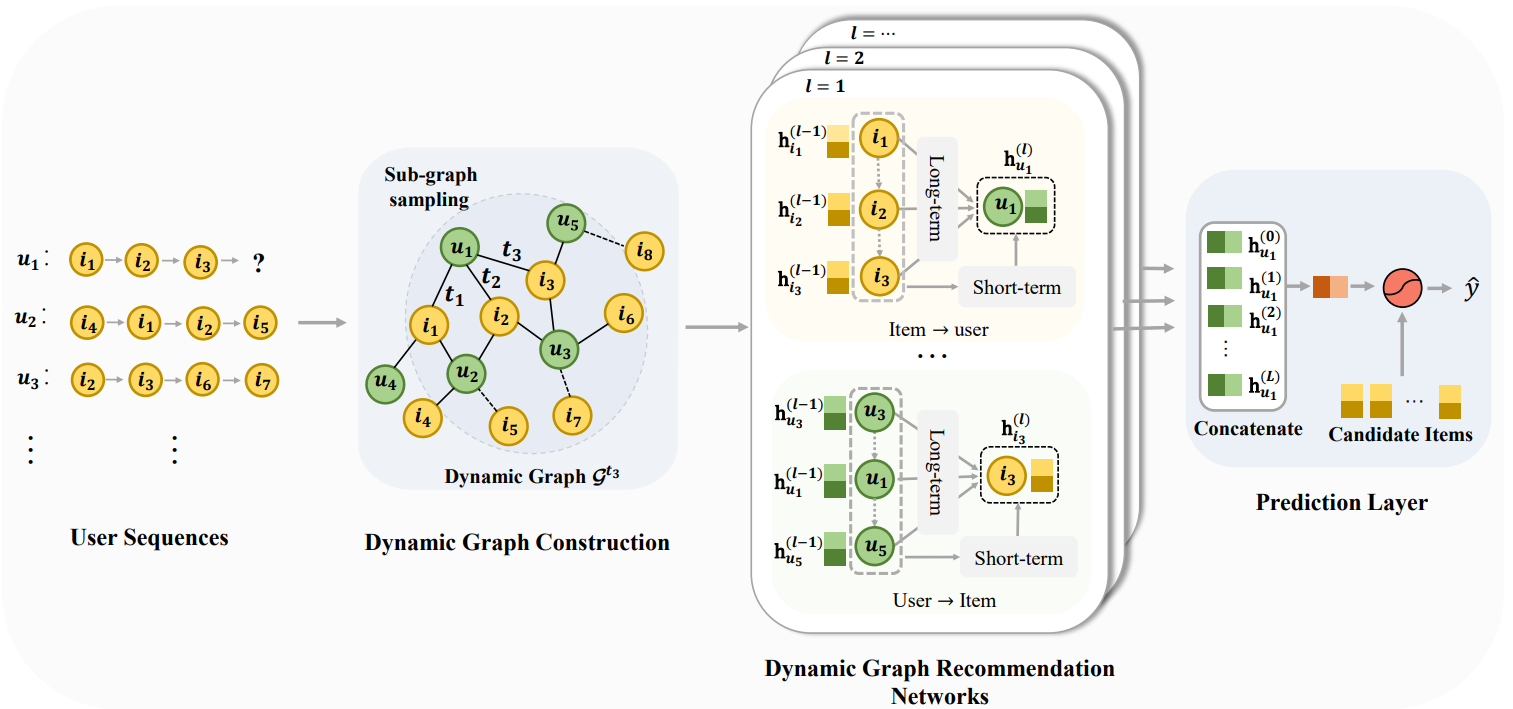
1. Dynamic graph construction
2. Sub graph sampling
3. Dynamic graph recommendation networks
4. Predication layer

Figure ‑ DSGR Architecture

The graph is constructed in the following manner, when a user u acts on an item i at time an edge is established between u and i, in addition to the interaction time between users and items the graph constructed also records the order information between them making the graph more suitable for the sequential recommendation task.  
Predicting the next item in the predicted sequence is equivalent to predicting which item is linked on the user’s node in the graph.

The next module in the architecture is the sub-graph sampling, as the number of the user sequence extends, the number of neighboring sequences increases and the scale of the dynamic graph composed of all users is also gradually expanding, all this increases the computational cost and introduces more noise to the graph to handle this issue a sampling algorithm is used.

A user node is chosen as the anchor node and select its most recent n first order neighbors from the graph where n is the maximum length of user sequence, for each item we use them as anchor node to sample the set of users who have interacted with them.

After sampling each sub-graph contains the nodes of the sequence and its associated sequences.

In the DGRN component consists of message propagation and node updating components, the message propagation mechanism aims to learn the information to propagate from user to item and from item to user.

Predicting the next item for a user is equivalent to predicting the link of user node u of a subgraph.

After acting the L layers on DGRN on graph, we obtain L embeddings of the user’s node, the user’s embedding in each layer emphasized various user preferences, these embeddings are concatenated to get final embedding for node.

To get score of each item a link function is defined as

Where is the user’s concatenated embedding and trainable transformation matrix.

The score indicates the confidence that the user will next try a given item.

## 3.7. Learning to Measure Changes: Fully Convolutional Siamese Networks for Scene Change Detection[11]

The paper aims to solve the challenge of scene change detection in a noisy environment, noise is generated by varying illumination, shadow and difference in camera viewpoints.

The method proposed aims to compare directly the dissimilarities between a pair of features using a Convolutional Siamese metric network to measure this change.

A Siamese network is a class of neural network architectures that contain two or more identical subnetworks, meaning they have the same parameters and weights and parameter updating is mirrored across both sub-networks.[12]

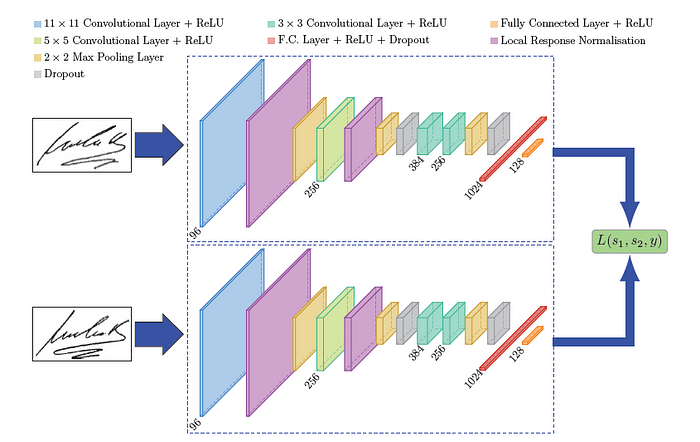


Figure ‑ Example Siamese network

The proposed model takes as input a raw image pair to generate a feature pair then uses a distance metric (Euclidian distance or cosine similarity) to produce change map.

The change map indicates how much confidence applies to the changes, the model aims to customize an appropriate metric to obtain higher distance value of changed pairs and a lower distance for unchanged pairs, to achieve this a contrastive loss was used to pull together unchanged pairs and push apart changed pairs.

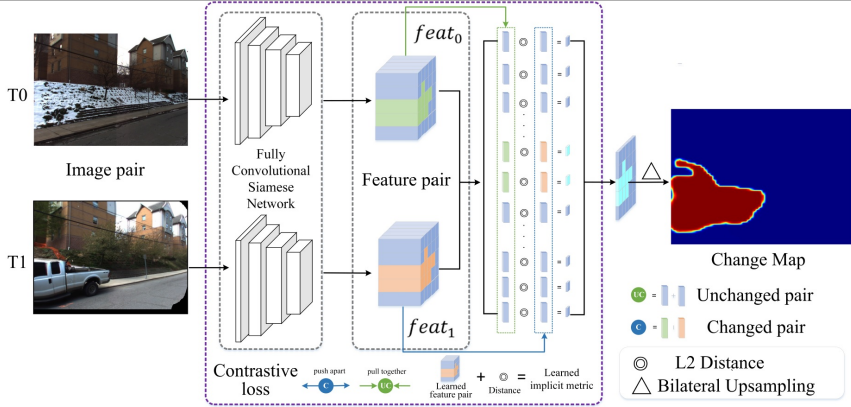
The Siamese network used to learn more discriminative features of the images; it maps image pairs to feature pairs.

Figure ‑ CosimeNet

A contrastive loss is used to produce high value for changed pairs and lower value for an unchanged pair, the loss is formulated as:

This loss aims to enlarge interclass difference and reduce interclass variation simultaneously.

## 3.8. Key-Frame Extraction Based on Histogram and Adaptive Clustering[13]

Extracting key frames from video has been recognized as one of the important research issues in video information retrieval.

The paper aims to do key-frame extraction using clustering as video key-frame is a relatively subjective concept and there is no unified criterion for evaluating the quality of keyframes, so using an unsupervised clustering can combine the characteristics of video content well.

The video is sampled to frames where each frame is then used to calculate a histogram feature.

Each frame is divided to a grid of 3x3 blocks and for each block, color histograms are created separately for the three RGB channels. Each histogram uses 6 bins, resulting in a total of 6 bins \* 3 channels = 18 bins per block. Therefore, each frame in the video is represented by a 1944-dimensional feature vector (9 blocks \* 18 bins = 162 bins for each channel), capturing the color distribution information.

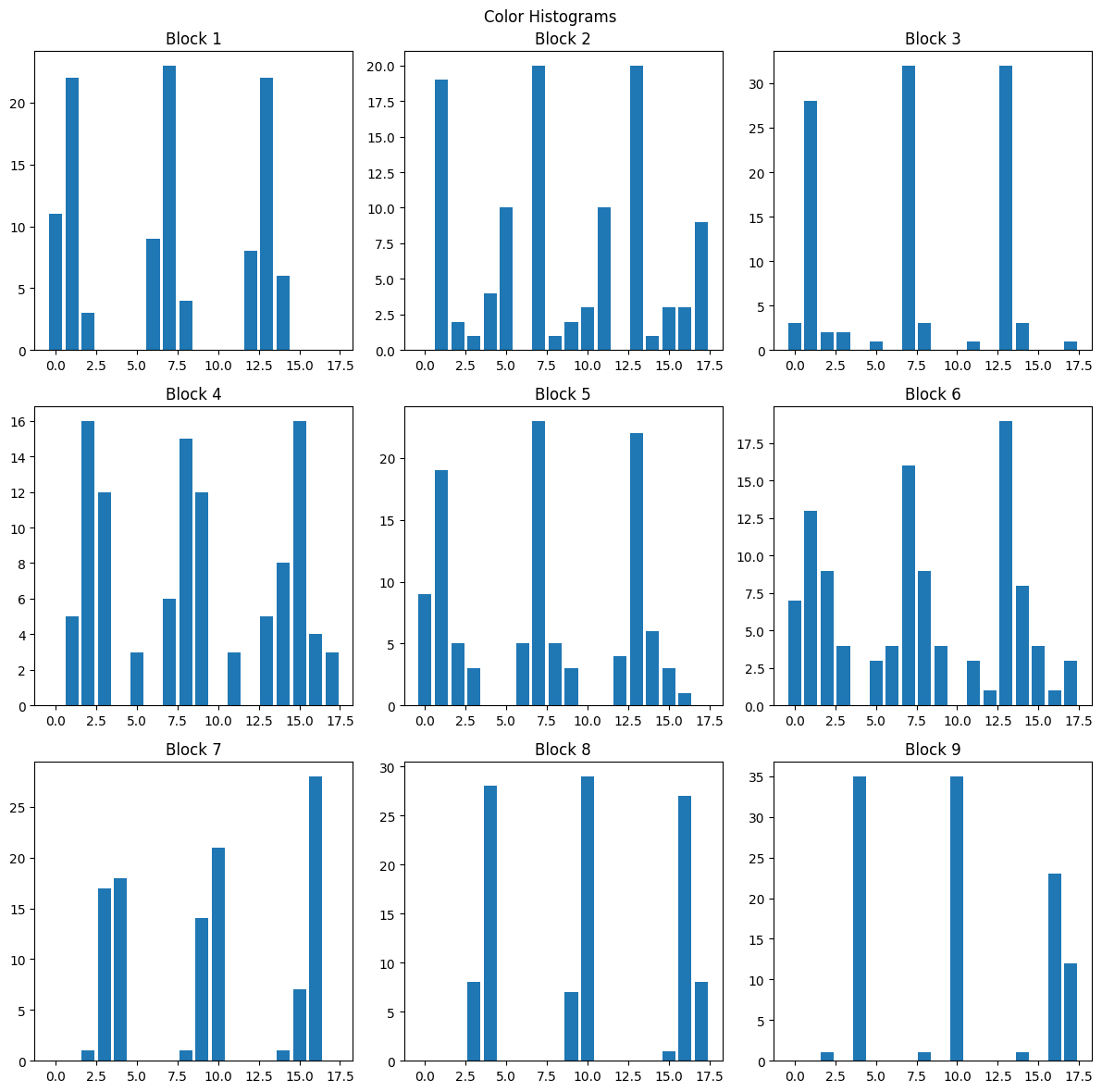
Then all the feature vectors are concatenated to form a feature-frame matrix.

Figure ‑ Example frame

Figure ‑ Frame as block

Each subplot represents a block in the 3x3 grid of the video. Each subplot represents the color histograms for that particular block.

Then dimension reduction is employed using Singular Value Decomposition, SVD captures the most important significant information while reducing the dimensions.

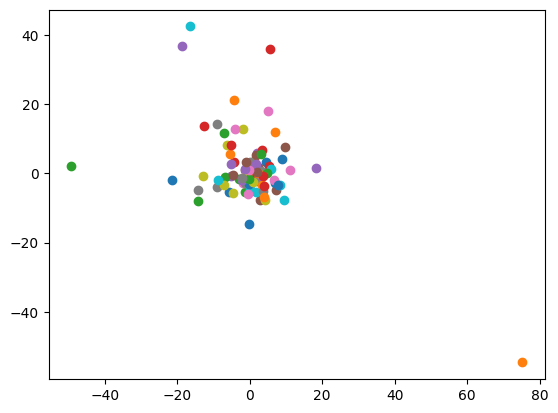
 The output of SVD is then clustered using dynamic clustering to group consecutive frames into cluster based on similarity.

Figure ‑ Centroids of clusters formed

The process of key-frame extraction involves the identification of transition frames and the selection of key-frames from dense clusters. Transition frames, found within sparse clusters, signify scene changes or significant alterations in the video. These frames are disregarded as they capture the moments of transition between shots rather than representing the essence of a specific shot.

On the other hand, key-frames are selected from dense clusters, which consist of frames that exhibit similarity and belong to the same shot. By grouping these frames together, the dense clusters effectively capture the essential content of a particular shot. To identify the key-frame for each shot, the last added frame within the dense cluster is chosen. This selection process ensures that the chosen key-frame represents the essence of the entire shot, condensing the visual information into a single representative frame.

By considering both the exclusion of transition frames and the selection of key-frames from dense clusters, this method aims to extract key-frames that best summarize the content of a video while excluding transitional moments and focusing on the most representative frames within each shot.

## 3.9. Comparative Study of Previous Work

We presented 2 recommender system architectures and we compared both to find a solution that fits our requirements.

We looked at the data needed, complexity, accuracy and real time performance.

Table ‑ DSGR vs Two-Tower

|  |  |  |
| --- | --- | --- |
| Comparison | DSGR | Two-Tower(eBay) |
| Data needed | Requires sequential data, such as user interactions or behaviors over time. It relies on capturing the temporal dynamics and patterns in user actions to make recommendations. This may include data such as clickstreams, purchase history, or browsing behavior. | Relies on user-item interaction data, which includes user preferences and item attributes. It focuses on capturing user-item interactions, such as ratings or explicit feedback, to generate recommendations. This approach may not heavily depend on sequential information but rather focuses on static user-item interactions. |
| Complexity | Complex due to the need to model and analyze the dynamic temporal dependencies between user actions. Building and updating sequential graphs, handling evolving user behaviors, and incorporating time dynamics can introduce additional complexity to the recommendation system | Less complex compared to the dynamic sequential graph approach. It involves training separate towers, i.e., neural networks, for user and item representations and learning their interaction. While it still requires careful feature engineering and model design, the overall complexity is typically lower than dynamic sequential graph recommendations. |
| Accuracy | The use of temporal dynamics and sequential patterns in user behavior can enhance the accuracy of recommendations, particularly in scenarios where user preferences change over time or have strong sequential dependencies. By capturing the evolving user interests, this approach can potentially provide more personalized and relevant recommendations. | While it may not directly leverage  sequential information, the two-tower system can still provide accurate recommendations by capturing user preferences and item attributes. The focus is on modeling the user-item interactions and learning effective representations to generate accurate recommendations. However, it may not capture the temporal dynamics and evolving interests as effectively as dynamic sequential graph recommendations. |
| Real-time performance | Real-time performance can be a challenge for dynamic sequential graph recommendations, especially in scenarios with large-scale data or high-speed data streams. Processing and updating sequential graphs in real-time can be computationally intensive and may require efficient algorithms or distributed systems to handle the incoming data and provide recommendations with minimal delay. | This system generally lends itself well to real-time performance.  The two-tower architecture allows for efficient computation of user-item interactions and can handle online updates of recommendations. With appropriate infrastructure and optimization, this approach can provide recommendations in real-time or near real-time, making it suitable for dynamic environments where timely recommendations are crucial. |

For scene change detection we had 2 options either to use Siamese Neural Network to find change between each pair of frames or use keyframe extraction.

We also looked at the same metrics to get a clear idea of which method we would base our solution on.

Table ‑ CosimeNet vs Keyframe extraction

|  |  |  |
| --- | --- | --- |
| Comparison | CosimeNet | Keyframe extraction |
| Data needed | Requires labeled pairs of frames indicating whether they belong to the same or different scenes. Training the siamese network relies on having a dataset with scene change annotations or manually creating such annotations for training purposes. | Relies on video data without the need for explicit annotations indicating scene changes. It utilizes color histograms and clustering techniques to identify key-frames based on the similarity of frames within shots. |
| Complexity | Implementing fully convolutional siamese networks can involve complex network architectures and training procedures. Fine-tuning deep neural networks and handling large amounts of training data can add to the complexity of this approach. | This method is generally less complex compared to using siamese networks. It involves feature extraction using color histograms and applying clustering algorithms to group frames into shots. The overall complexity is relatively lower compared to deep learning-based approaches. |
| Accuracy | Siamese networks can learn to measure changes between frames and are capable of providing accurate scene change detection results. The approach benefits from the ability of deep learning models to capture complex patterns and dependencies in the data. | The accuracy of key-frame extraction using histogram-based methods and adaptive clustering depends on the effectiveness of color histograms in capturing the key characteristics of shots and the performance of the clustering algorithm. While this approach can yield satisfactory results, it may not achieve the same level of accuracy as deep learning-based methods. |
| Type | Deep Neural Network | Classical Machine learning algorithms |

## 3.10. Implemented Approach

To implement our system, we reviewed multiple papers, methods and architectures to find what best fit our requirements.

We quickly settled on using BERT as our chatbot model because of we can use a pretrained BERT that has been trained on a large amount of Arabic text data, which allows it to capture and understand the nuances of the Arabic language. Leveraging a pretrained model like BERT can save time and resources compared to training a language model from scratch.

Also, BERT is easily fine-tuned on specific tasks so we used it as our backbone in the chatbot module.

DSGR was selected for its unique capability to capture and adapt to temporal changes in user preferences. By utilizing sequential data and implicit feedback, DSGR can dynamically model the evolving interests and preferences of users, enabling the system to deliver personalized and up-to-date recommendations that align with their changing needs. Unlike the Two Tower Recommendation System, which relies on explicit ratings, DSGR effectively handles implicit feedback, such as clicks and views, making it suitable for scenarios where explicit ratings are unavailable. Additionally, DSGR's temporal adaptation helps overcome the cold-start problem by analyzing early user interactions and tailoring recommendations even in the absence of extensive user history.

For scene change detection method we weren’t sure which method would better fit our use cases, so we implemented both and found that CosimeNet performed poorly in our use case as it was trained on detecting change between street view image pairs which made it hard for the model to detect change in smaller context, and there’s no suitable dataset that matches our use case.

While extracting keyframes proved more successful in detecting scene changes in the recipes video.

# System Design and Architecture

In this chapter we’ll walk through the system design and the building process of our project.

The project has 4 main modules: chatbot, recipe recommender and recipe video division, the first 2 modules are tightly coupled and designed to work together to provide the experience intended.

We’ll talk in details about each module and how they are used in the context of our application.

## 4.1. Overview and Assumptions

We made some important assumptions that enabled us to create the best experience possible with the number of constraints employed.

The first assumption we made was that the chatbot wasn’t a general use cooking chatbot that can answer any question you have about recipes, it’s a chatbot designed to only understand and reply to specific intents (that will be explained in the chatbot module).

The second assumption was that there are multiple recipes that are almost identical, since there was no ready dataset that includes recipes in Arabic, we had to collect recipes, the recipes were submitted by users in the original website meaning we get multiple versions of the same recipes and automatic detection was inconsistent.

The third assumption made is that not all recipes have videos that explain the recipe, so we decided to use Youtube’s search API to get short videos that best match the recipe name and use that video as the recipes video to be analyzed, the fourth assumption is also related to the video analysis, the videos used will be videos that are shot focusing on the cooking process itself.

These assumptions helped us to focus on the core functionalities of the system.

## 4.2. System Architecture

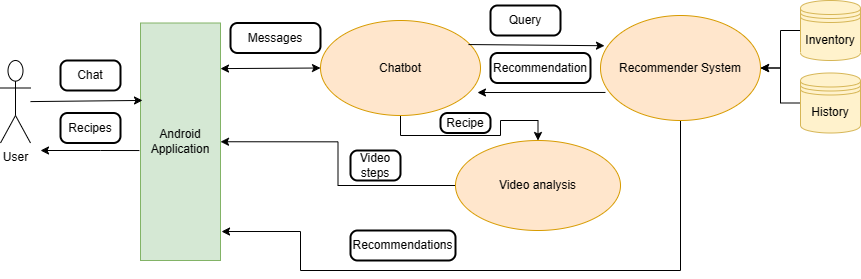
We’ll introduce the architecture of Chef&1/2 and go over each of its modules and how they all interact to form the application.

Figure ‑ Chef&1/2 high level architecture

### 4.2.1. Block Diagram

In our system architecture, the user interacts with the Android application, which provides a user-friendly interface focused on delivering recipe recommendations and quick access to the chatbot for personalized cravings. Let's break down the flow and interactions between the different components:

1. Recipe Recommendations:

Upon launching the Android application, the user is presented with recipe recommendations provided by the recommender system. The user can browse through the recommended recipes and select any of them based on their interests or preferences. These recommendations are generated by the recommender system, taking into account the user's taste, history, and possibly other factors such as popularity.

2. Chatbot Interaction:

The user has the option to engage with the chatbot, which is accessible within the application. The chatbot provides a chat-like interface where the user can communicate and ask for recipes that fit their specific cravings. The chatbot processes the user's message, creating a query to be sent to the recommendation system. The query considers the user's preferences, taste, and history, and aims to retrieve recipe recommendations that closely align with the user's request.

3. Recipe Presentation and Feedback:

The chatbot begins presenting recipe recommendations to the user one by one based on the query results from the recommendation system. The user has the option to provide feedback on each presented recipe. They can accept the recipe if it matches their preferences or request another recipe if it doesn't meet their requirements. This back and forth continues for a few iterations until the user settles on a recipe that they find suitable and appealing.

4. Recipe Details and Video Analysis:

Once the user selects a recipe, they are redirected to the recipe's details page. Here, they can find comprehensive information about the recipe, including ingredients, instructions, and other relevant details. The recipe's video is also processed using video analysis techniques to produce timestamps that divide the video into steps. This division facilitates navigation and allows users to easily follow along with the recipe video.

5. Data Flow and API Integration:

The modules in the system interact with each other by obtaining the necessary data from an API connected to the database. The modules process the inputs, such as user messages, queries, and recipe details, and send the results through an API to the Android application. This architecture allows for the possibility of running more complex models remotely on desktops and sending the processed results to the application, ensuring efficient and effective handling of data and computational resources.

By integrating the various components of the system, including the recommender system, chatbot, recipe details, and video analysis, the application provides users with a seamless experience, allowing them to discover and explore recipe recommendations tailored to their taste while also accommodating their specific cravings through interactive chatbot interactions.

## 4.3. Module 1: Chatbot

This section is dedicated to explaining the details of the chatbot module, we’ll detail its function and its components and the constraints that affected its design.

### 4.3.1. Functional Description

In our application architecture, the chatbot takes on a crucial role in interacting with users and catering to their specific cravings. While the daily recommendations provided by the recommender system are accurate in depicting what users should eat, there are times when users have a strong desire for a very specific dish. This is where the chatbot becomes essential.

The primary function of the chatbot is to understand the user's craving and generate queries that the recommender system can process. The chatbot engages in a conversation with the user, gathering information about their specific desires, such as the type of dish, specific ingredients, cuisine preferences, or any other relevant details. Based on this information, the chatbot formulates queries that are sent to the recommender system to retrieve suitable recipe recommendations.

Once the recommender system provides a list of recipe recommendations, the chatbot presents them to the user one by one, waiting for the user's feedback on each presented recipe. The chatbot takes into account the user's feedback to understand their preferences better and can adjust the query during the feedback phase to generate a new set of recipe recommendations.

Overall, the integration of the chatbot within your application architecture enhances the user experience by allowing users to express their specific cravings and receive personalized recipe recommendations. The chatbot serves as an interactive and adaptable interface that bridges the gap between user desires and the recommendations provided by the recommender system.

### 4.3.2. Modular Decomposition

The chatbot module is composed of 4 parts: intent classifier, named entity recognition and query formatter all managed by state manager.

While BERT is a strong NLP model that can be fine-tuned for use a chatbot on its own that can handle long conversations, the lack of data that fit the scenario of asking for recommendations and handling feedback we decided to user BERT as text classifier to predict the intent of the message and add another NLP model that extract important entities in the user’s message.

After considering multiple methods we settled on using intent based chatbot, the BERT model was fine-tuned to classify the intent of the user’s messages, we are interested in finding out 3 intents that perfectly fit the use case of the chatbot.

When the user engages with the chatbot he would normally either ask for a recommendation, accepts the recipe recommended or refuses the recipe. These 3 intents are enough to guide the user from the start where they are not sure what to eat until they pick a recipe.

Table ‑ Intent example from own survey[14]

|  |  |  |  |
| --- | --- | --- | --- |
| Intent | Ask | Accept | Refuse |
| Example | عايز اكلة شبه النجرسكو | تمام حلوة الوصفة دي | لا شوف حاجة تانية |

Intent classifier’s job in the chatting loop is to figure out what the user’s intent to be able to form a query and move from the asking phase to the recommendation phase.

Figure ‑ Chat interaction with the intent Ask

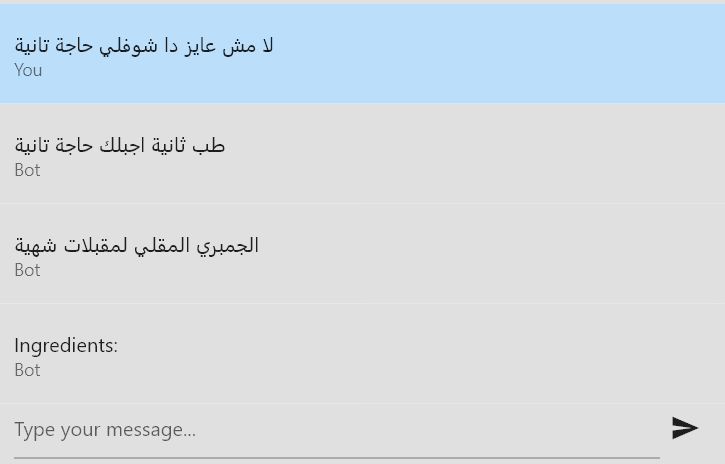


Figure ‑ Chat interaction with the intent Refuse

The second module inside the chatbot is the Named Entity Recognition, Named Entity Recognition (NER) is a natural language processing task that involves identifying and classifying named entities in text into predefined categories such as names of people, organizations, locations, dates, and more. NER algorithms aim to extract and label these entities, enabling machines to understand the context and meaning of text. By automatically identifying and categorizing named entities, NER plays a crucial role in various applications such as information retrieval, question answering, text summarization, and sentiment analysis, enhancing the overall understanding and analysis of text data. [15]

Figure ‑ NER example

Chef&1/2’s NER is trained to detect 3 types of entities: recipe names, ingredients and specific tags that are used to describe the recipes.

Table ‑ NER model examples

|  |  |
| --- | --- |
| Phrase | Entities |
| عايز وصفة أكل سريعة و سهلة عشان مستعجل | (14, 19, 'Tag'), (22, 25, 'Tag') |
| عايز اكلة شبه النجرسكو | (14, 22, 'Recipe') |

The intent along with the entities extracted from the user’s message are fed into the query formatter to produce a query that is given to the recommender.

All the parts of the chatbot are managed by a state machine so the queries can be formed in a logical manner and move through the decision process correctly, we have 3 main phases: asking for recommendations, refusing the recommendation and finally accepting the recommendation.

The normal sequence followed is that the user first asks for a recipe, then receives one recommendation at a time where the user either accepts or denies the given recipe, the chatbot continues to query recommendations until the user is satisfied and accepts.

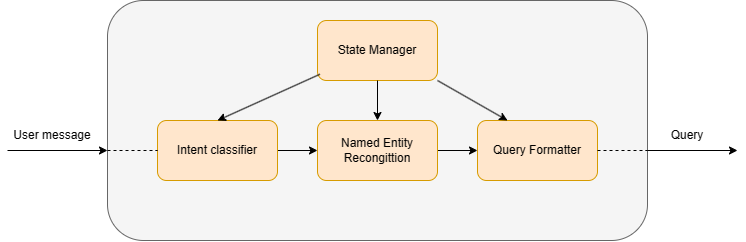


Figure ‑ Chef&1/2 Chatbot

### 4.3.3. Design Constraints

We slightly touched upon the constraints at the start of the module decomposition but we’ll expand more on it here.

We were heavily constrained by the limited conversation data that would fit our use case, leading to underutilizing BERT’s potential and we were forced to incorporate other models to support the designed intent classifier.

These constraints affected the overall performance of the chatbot as it now only serves the user in pre-determined ways.

## 4.4. Module 2: Recommender system

This section is dedicated to explaining the details of the recommender module, we’ll detail its function and its components and the constraints that affected its design.

### 4.4.1. Functional Description

In the context of our application, the recommender system has two primary roles: providing recipe recommendations on the user's homepage and fulfilling queries processed by the chatbot. Let's explore each role in more detail:

1. Homepage Recommendations:

The recommender system analyzes user preferences, historical behavior, and possibly other relevant data to decide which recipes to display on the user's homepage. It takes into account factors such as the user's taste, dietary restrictions, previous recipe interactions, and possibly external factors like popularity or trending recipes. By understanding the user's preferences, the recommender system aims to present recipes that are personalized and highly relevant to the user's individual taste.

2. Chatbot Query Fulfillment:

When a user interacts with the chatbot and requests a recipe recommendation, the recommender system plays a crucial role in finding the closest match to the user's request. It processes the user's query, which might include specific ingredients, cuisine preferences, dietary requirements, or any other relevant information. The recommender system leverages its understanding of the user's taste and history to filter through the available recipes and identify the most suitable recommendations. The results are then sorted based on the user's preferences and history, ensuring that the presented recipes align closely with the user's needs and interests.

### 4.4.2. Modular Decomposition

We have 2 types of recommendations, recommendations based on user’s history that are modeled using a graph neural network that produces recommendations using the user’s history and the history of similar users and recommendations based on specific asks that are inputted to the system through the chat interface.

To address the distinction between recommendations based on user history and those based on specific user queries, the recommender system consists of two recommendation modules: the Query Based Recommender and the DSGR (Graph Neural Network) module.

The Query Based Recommender module takes the user's query, formatted by the chatbot and containing relevant entities, as input. It begins by generating candidate recipes that match the provided query. This is achieved through the use of Elasticsearch, a powerful search engine known for its efficient querying and retrieval capabilities. Elasticsearch goes beyond simple keyword-based searches, offering advanced text analysis features such as tokenization, stemming, synonym expansion, and language-specific analyzers.

Table ‑ Elasticsearch results

|  |
| --- |
| Results for: بطاطس رز بصل |
| رز الأصفر بالبصل |
| بطاطس مقلية مع البصل |
| شوربة البطاطس بالبصل |
| بطاطس حلقات بالبصل |
| بصل مشوي بالبطاطس |
| الرز البخاري الأصلي |
| رز باللحمة والبطاطس والعدس |

Once the candidate recipes are obtained, the next step is to rank them based on the user's favorite recipes. This ranking process creates a curated list of recipes that is personalized to each user, taking into account their individual preferences and tastes.

Given that recipes are represented as text-based data, a model is employed to transform the recipe texts into meaningful vectors that capture relative meaning. This enables the recommender system to make effective comparisons between recipes and provide recommendations that align with the user's preferences.

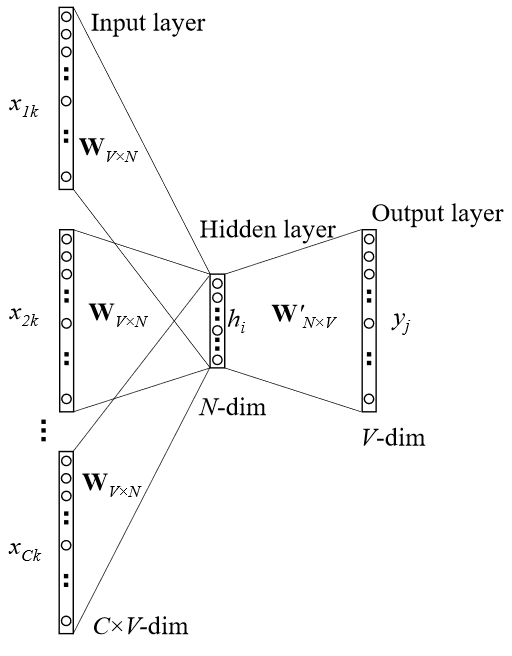
The first step was to choose the appropriate model that most fit our data and requirements.

Table ‑ Comparison of Text to vector models

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Description | Strengths | Weaknesses |
| Bag-of-Words (BoW) | Represents text as a collection of words | Simplicity, interpretability | Lacks context and word relationships |
| Word2Vec | Learns dense word embeddings | Captures semantic relationships, performs well on tasks | May struggle with rare words, limited by training data |
| BERT | Transformer-based model with contextual embeddings | Captures contextual meaning, versatile across NLP tasks | Requires substantial resources, data for training |
| N-gram | Considers sequences of N consecutive words | Captures local context and word order | Struggles with global context, longer-range dependencies |
| CBOW | Learns word embeddings based on context | Efficient training, performs well on semantic tasks | Less effective for syntax, word order, and longer dependencies |

We decided on using CBOW model as it efficient to train and fit the data that we had collected.

The model was trained on a corpus of 15911 recipes, each recipe consists of name, ingredients, tags and steps.



The model is capable of representing recipes in text to vectors in a vector space where we can measure similarity between each vector.

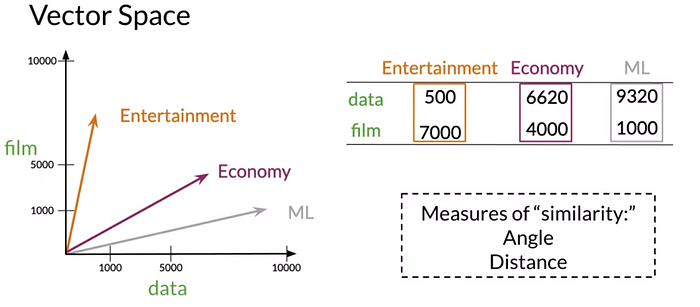
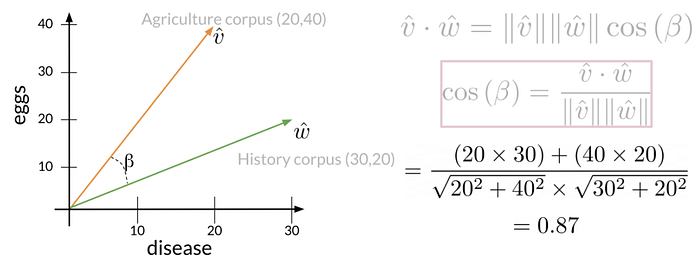


Figure ‑ Example of Vector Space[16]

We use the cosine similarity to find how similar are 2 recipe vectors, The main advantage of the cosine similarity is that it isn’t biased by the size difference between the representations.

The more similar 2 vectors are the more the cosine value is to 1, the farther they are apart the value is closer to 0.

This helps in finding top k nearest recipes and measuring the similarity of two recipes.

Table ‑ Most similar to example

|  |
| --- |
| Nearest to مهلبية البطاطا الحلوة |
| فيديو كيكة جوز الهند الهشة |
| سلطة البطاطا الحلوة |
| كرات جوز الهند بالشوكولاتة البيضاء واللوز |
| مهلبية الشوكولاتة بجوز الهند |
| مهلبية الشوفان بجوز الهند |

Each candidate recipe is presented by a vector and all the user’s favorite recipes are averaged into one recipe vector, and we rank candidate recipes based on how similar they are to the user’s taste.

On the other the DSGR model handles producing personalized recommendations based on past interactions and behaviors.

The module starts by constructing a user-recipe interaction graph. Each user is represented as a node in the graph, and each recipe that the user has interacted with (cooked) is also represented as a node.

Edges are added to connect users and the recipes they have interacted with, representing the interactions between them. These edges capture the user's history of recipe interactions.

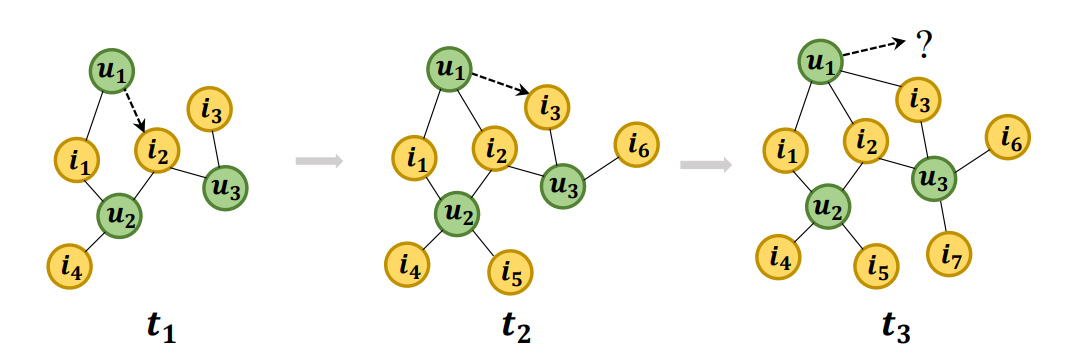


Figure ‑ Graph construction

Once the interaction graph is constructed, a graph neural network (GNN) is trained on this graph.

The GNN learns to aggregate information from neighboring nodes in the graph, capturing the patterns and relationships within the user-recipe interaction graph.

After the GNN is trained, it is used to generate personalized recommendations for users.

Given a target user, the GNN leverages the learned representations and the graph structure to infer the user's preferences and identify recipes that are likely to be of interest.

The recommendations are generated based on a combination of the user's historical interactions and the preferences of similar users in the graph. By leveraging the graph structure, the model can capture collaborative filtering effects, where recommendations are influenced by the behavior of similar users.

Predicting the next recipe that the user would make is equivalent to predicting is there and edge connecting the user’s node and the item’s node.

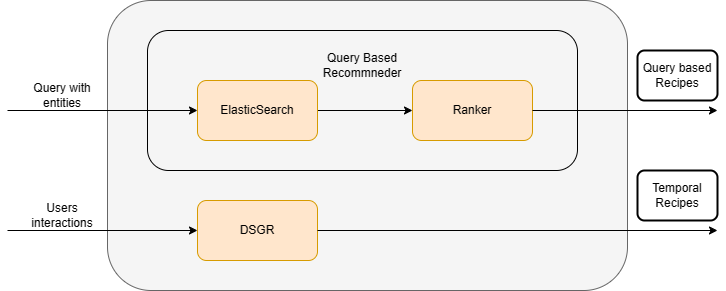


Figure ‑ Chef&1/2 Recommender system

### 4.4.3. Design Constraints

The recommender is constrained by the available recipes in the system, the recommender can’t mix and match in the recipes to provide unique recipes.

The cold start problem refers to the challenge of providing accurate and relevant recommendations for new users who have limited or no interaction history within the system. When a user first joins the application, there is insufficient data available to understand their preferences and make personalized recommendations. This constraint poses a difficulty in accurately tailoring recommendations to the specific tastes and preferences of new users.

Another related constraint is the low number of interactions. For users who have recently joined the platform or have not actively engaged with the system, the available data on their interactions may be limited. This low number of interactions reduces the amount of information available to the recommender system for understanding user preferences and generating accurate recommendations. Without a substantial history of interactions, it becomes challenging to identify patterns and preferences that can drive effective recommendation generation.

To combat these constraints, we model the behavior of new users based on the early interactions of existing users. By leveraging the data used to train the network, we can infer similarities and patterns among users and utilize this information to provide recommendations for new users.

When a new user joins the system, their lack of interaction data poses a challenge in understanding their preferences. However, by analyzing the early interactions of existing users, we can identify commonalities and similarities in their behavior during the initial stages of their engagement with the platform.

By doing so, we can make educated assumptions about their preferences and generate initial recommendations. This approach leverages the knowledge gained from existing users to provide relevant suggestions to new users, even in the absence of their own interaction history.

## 4.5. Module 3: Video analysis

We’ll talk about the last main module in our application, the video analysis module that is responsible for analyzing the video recipe to steps, we’ll detail its function and its components and the constraints that affected its design.

### 4.5.1 Functional description

The video analysis module plays a crucial role in the final part of the application's main flow. Once the user selects a recipe to make, they are presented with a corresponding video. To enhance the user experience and facilitate navigation within the video, the video analysis module is responsible for analyzing and dividing the video into distinct steps.

The video analysis process begins by retrieving the recipe's video from YouTube. Next, the module performs keyframe extraction, which involves selecting a set of frames that best capture the essential content of the video. Each extracted frame is then fed into a ResNet34 model, a deep neural network architecture, to extract a feature vector that represents the visual information contained in that frame.

Using these feature vectors, the video analysis module proceeds to identify which frames belong to the same scene and which frames indicate a scene change. This analysis allows for the identification of distinct steps within the video. By determining the boundaries between scenes, the module effectively divides the video into separate segments, each representing a specific step in the recipe.

This scene-based division enables users to navigate through the video more easily, providing a seamless and intuitive experience. Users can jump to specific steps within the video, replay certain portions, or skip to the next step without having to manually scrub through the entire video.

By leveraging keyframe extraction, feature extraction using the ResNet34 model, and scene detection algorithms, the video analysis module enhances the usability of the application's video content. It enables users to effectively follow along with the recipe's instructions, ensuring a smooth and efficient cooking experience.

### 4.5.2 Modular Decomposition

The Video analysis module consists of 4 smaller modules that each play a role in creating the video analysis pipeline.

The first thing we need to get the recipe’s video, we use youtube’s search API to get the video that matches the recipe chosen, we then download it to the device for playback and download to the API host machine to begin processing.

The video is then passed on to Keyframe extraction module, the module first divided the video into a grid of 3x3 blocks, and for each block, color histograms are created separately for the three RGB channels. Each histogram uses 6 bins, resulting in a total of 216 bins per block. Therefore, each frame in the video is represented by a 1944-dimensional feature vector, capturing the color distribution information.

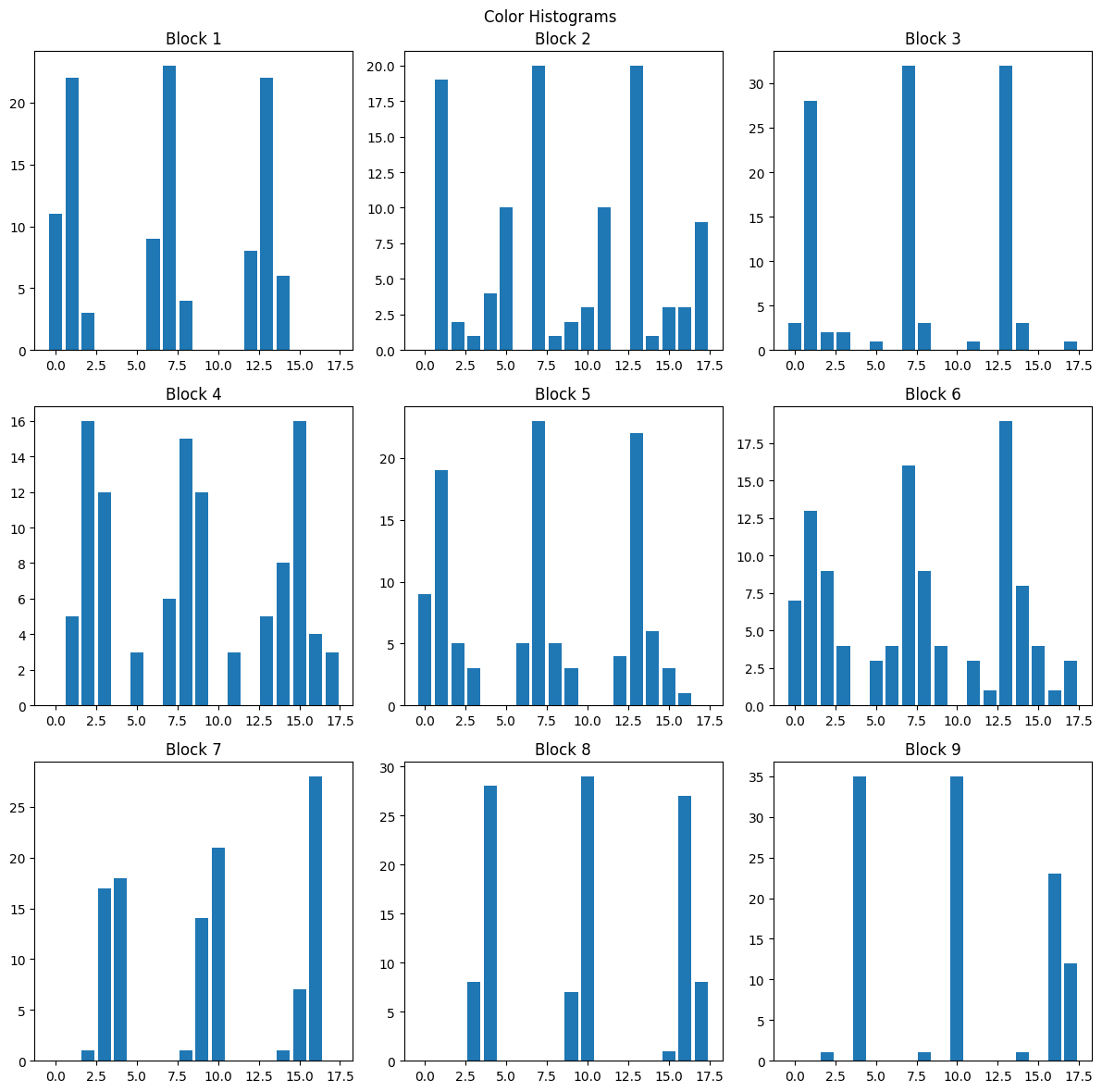


Figure ‑ Color histograms for a given frame

We repeat the same process for all the frames in the video and each vector is added to a matrix, the matrix becomes of dimension .  
The next step is to reduce the dimension using Singular Value Decomposition,

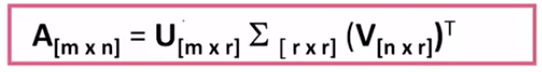
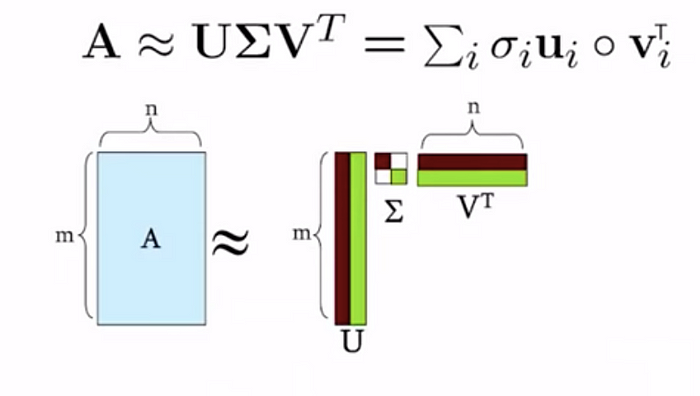
The idea behind SVD is to take this matrix A of order m x n and represent it as a product of three matrices, further these three matrices would be having certain constraints associated with it.[17]

Figure ‑ SVD Graphically

Dimensionality reduction techniques, such as Singular Value Decomposition (SVD), offer several benefits. They help in discovering hidden correlations within data by identifying latent dimensions along which the data varies. These techniques also enable the removal of redundant and noisy features, reducing the dimensionality while preserving influential features. By transforming high-dimensional data into lower-dimensional representations, interpretation, visualization, and analysis become easier. Furthermore, dimensionality reduction aids in the storage, processing, and analysis of data by reducing its size and improving computational efficiency.

After applying SVD the dimension is reduced to , then we can start the next step to dynamically cluster the frames according its corresponding feature vector.

The dynamic clustering method is employed to group consecutive frames into clusters based on their similarity. The cosine similarity measure is used to compare the new frame to the last formed cluster. If the new frame is similar to the last cluster, it is added to that cluster; otherwise, a new cluster is created.

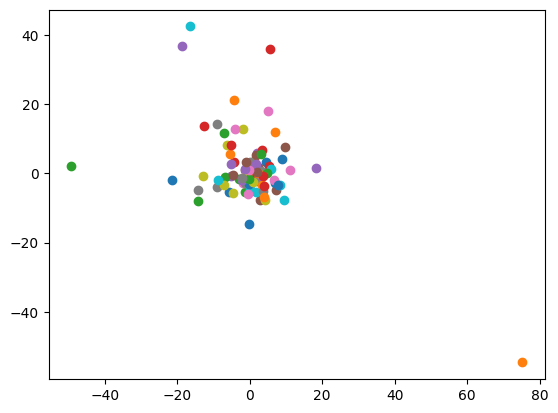
Dense clusters, which encompass frames that share similarities and belong to the same shot, are identified and selected. From each dense cluster, the last added frame is chosen as the key-frame, capturing the essence of that particular shot.

Figure ‑ Clusters formed

We then take each frame and save it to be used in the next phase of the scene change detection process.

Each frame is preprocessed to be used as input to a ResNet34 model to extract image features, the image feature vectors are then passed to the last module the scene change detection module.

The Scene Change Detection uses k-means clustering to cluster the frame according their feature vectors generated by ResNet34, we use K to be the number of the steps in the recipe.

Since the frames are sorted by the order that they appear in the original video we observe if the cluster that the frame belongs to changes with time or not to detect if there has been a scene change or not.

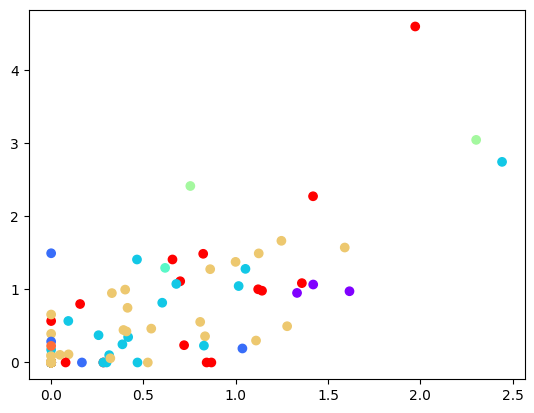


Figure ‑ Frames Clustered using KMeans

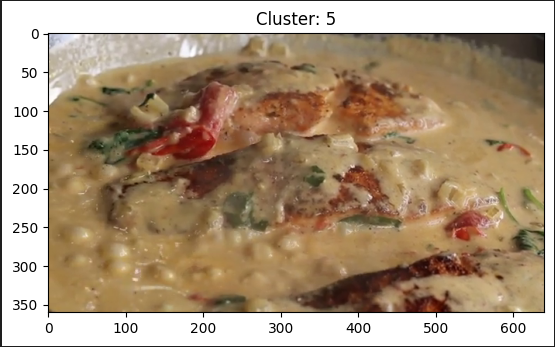
****

Figure ‑ Scene Change Detection Example

Time

With the changing of time if the cluster that the frame belongs to stays the same then it’s the same scene and there’s no need to introduce a scene change here.



Figure ‑ Scene Change Detection Example

Time

With the changing of time if the cluster that the frame belongs to changes then there was a scene change and we introduce a new scene at beginning of the frame.

The module goes through all the keyframes extracted and repeats this process to get the frames that belong to the same scene, then it outputs the timestamps at which every new scene is introduces to be used by a video player for navigation.

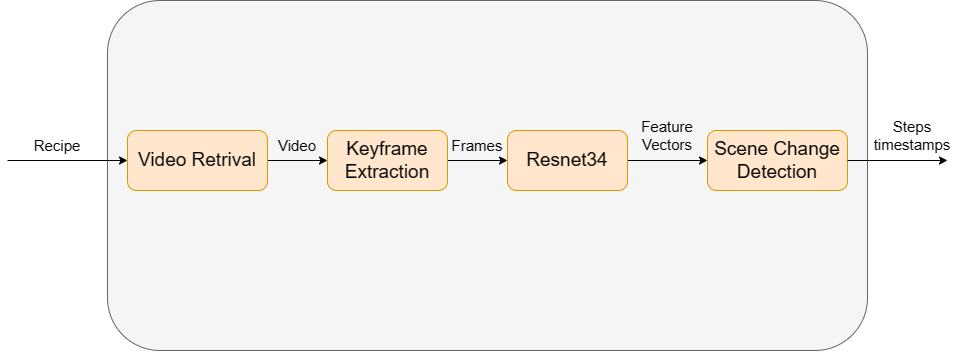


Figure ‑ Chef&1/2's Video analysis module

## 4.5.3. Design Constraints

Since we plan to do all this analysis on demand at least once for each new recipe, the process needs to be as fast as possible but the operations that are involved take some time to process, that’s why the video retrieval get short videos that aren’t more than 5 minutes long.

We can process longer recipes video ahead of time but the online process is limited to shorter videos.

We also consider that all recipe videos would be in a specific video format that shoots the cooking process using close up shots that focus solely on the cooking surface, while other video formats, would work this process would produce a lot more cuts and unnecessary steps.

# System Testing and Verification

Testing has become an important part of any software development; it helps to verify that the module performs its task successfully and behaves correctly in both ideal conditions and failures.

Especially with a software that contains multiple modules that rely on one another to provide the full experience, it is imperative that each module is tested and verified before we can test the whole system as a unit.

We’ll discuss the testing method used in each module during this chapter.

The metrics on which we evaluated vary from one module to another and we will go into the details while discussing each module’s testing.

First, we will explain the testing setup and then present our plan and strategy during the testing phase finally we’ll compare our results with the results from previous work if there was any available.

## 5.1. Testing Setup

A demo version of the application was developed to carry on the testing of each module, the demo version had all 15911 recipes indexed and computed each recipe’s vector representation.

The basis of the application that was needed to enable testing other modules, it is the recommender system, we created a testing user that we assigned to 5 favorite recipes and create a small history of interactions.

The application also had a simple interface that would accept text and would show the intent detected.

For the video analysis module, we created a separate python program that executes the analysis on the video and then play the video with the option to jump through the steps.

## 5.2. Testing Plan and Strategy

Each module is tested individually before integrating the system together, each module was to be tested with data that would best mimic real data that was going to be normally used.

We wanted to make sure that each module when given the right input, produces the desired result as to uncover any bugs or unseen problems, prioritizing the existing features, we can concentrate on validating the intended behavior and verifying that the modules perform their tasks correctly. This approach allows us to detect and address any unexpected outcomes, edge cases, or errors that may arise.

We Identify the different levels of testing to be conducted to be unit testing and integration testing.

### 5.2.1. Module Testing

We’ll go over how each module was tested, which data was used and how did the system perform.

### 5.2.1.1 Recommender System

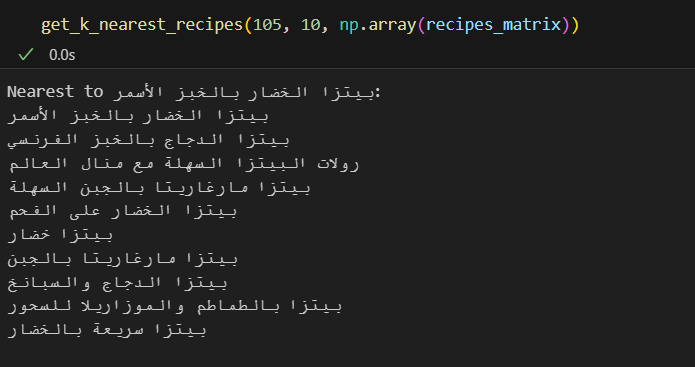
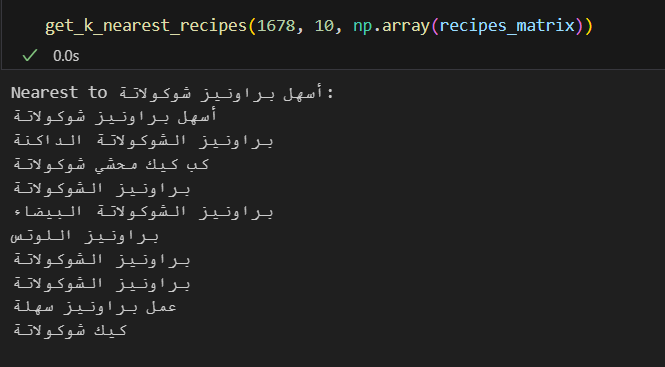
Since the recommender system is comprised of 2 modules, we tested each using a different method, for the query-based recommender we tested how our CBOW model represented different recipes by running get top k nearest on multiple recipes.

Figure ‑ Example of k nearest recipes

From the results of testing, we can see the CBOW model embeds the recipes quite well, and captures the different characteristics of the recipes.

This gave us confidence of using the recipes vector resulting from the model to find similar recipes or rank recipes.

As for the DSGR module, its recommendations are based of extensive history of the user interactions, to properly its performance we used offline evaluation after the training of the network is completed.

NDCG@N (Normalized Discounted Cumulative Gain at N) and Hit@N are evaluation metrics commonly used in information retrieval and recommendation systems to assess the quality of ranked lists or recommendations.

We used NDCG@20, Hit@20, NDCG@10 and Hit@10 as metrics to evaluate and test the performance of our DSGR.[18]

NDCG@20 measures the effectiveness of a ranking or recommendation list by considering both the relevance and the position of the items. It calculates the cumulative gain of the top 20 items in the list, discounted based on their position. The gain represents the relevance or utility of the item. NDCG@20 normalizes the cumulative gain by dividing it by the ideal DCG (Discounted Cumulative Gain) at 20, which is the maximum achievable DCG. This normalization ensures that the NDCG@20 score falls between 0 and 1, with higher values indicating better rankings or recommendations.

Hit@20, on the other hand, is a binary metric that measures whether the ground truth or relevant items are present in the top 20 positions of a ranked list or recommendations. It simply checks if at least one relevant item is included in the top 20. A hit is considered when there is a match, and the Hit@20 score is the percentage of ranked lists or recommendations that achieve at least one hit.

Similarly, NDCG@10 (Normalized Discounted Cumulative Gain at 10) and Hit@10 are variants of the above metrics but calculated based on the top 10 positions instead of 20.

Table ‑ DSGR evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| NDCG@20 | Hit@20 | NDCG@10 | Hit@10 |
| 0.3302 | 0.6109 | 0.2958 | 0.4747 |

Looking at the Hit metrics, we observe that out of a total of 15,911 recipes that were considered, 61% of the recipes that the user would potentially choose to make next are included in the top 20 highest scoring items.

The same can be observed for the Hit@10 metric.

This finding indicates that the recommendation system or ranking algorithm has been successful in identifying and presenting a significant proportion of the recipes that align with the user's preferences.

### 5.2.1.2 Chatbot

Testing the chatbot's performance and ensuring accurate intent identification is crucial for its effectiveness. However, in cases where the data sample size is limited, creating a confusion matrix to comprehensively evaluate the chatbot's performance may be challenging.

To overcome this limitation, manual testing can be employed as an alternative approach. Manual testing involves interacting with the chatbot using various input queries and evaluating its responses based on the intended user intent. During this process, human evaluators play the role of users and assess how well the chatbot correctly identifies and understands their intents.

While manual testing may not provide a quantitative performance analysis like a confusion matrix, it offers qualitative insights into the chatbot's performance. Evaluators can provide feedback on the accuracy of intent identification, identify any misclassifications or misunderstandings, and suggest improvements to enhance the chatbot's performance.

To ensure reliable results, manual testing should involve a diverse set of evaluators, representing different user perspectives and using a range of test queries. Their feedback and observations can be used to iteratively refine the intent classification model and improve the chatbot's overall performance.

Although manual testing may be time-consuming and subject to evaluator bias, it serves as a valuable method for assessing intent classification performance when the data sample size is limited and a comprehensive confusion matrix cannot be constructed.

Table ‑ Chatbot examples

|  |  |
| --- | --- |
| Phrase | Intent |
| متقولي ناكل ايه النهاردا | Ask |
| ناكل ايه النهاردا, عايز حاجة فيها فراخ | Ask |
| لا دا مش حلو شوفلي حاجة تانية | Refuse |
| قولي حاجة اعملها | Agree (wrong detection) |
| هاتلي حاجة تانية | Refuse |
| ماشي حلو دا | Agree |

The intent classifier detects the intent correctly most of the time, but due to the limited data it doesn’t know many ways to express asking for a recipe.

### 5.2.1.3 Video Analysis

Given the subjective nature of video analysis, which can vary significantly from one video to another, manual testing played a crucial role in verifying the performance of the modules and ensuring they met our expectations.

To facilitate this testing process, we developed a dedicated testing application specifically designed for video analysis. This application takes a video as input and generates the corresponding step timestamps based on the analysis performed by the video analysis module. The testing application provides controls that allow users to skip to the next or previous steps, providing a user-friendly interface for navigating through the video and verifying the accuracy of the extracted timestamps.

The video player component within the testing application reads the output timestamps and exposes intuitive controls that enable users to navigate through the video content based on the provided steps. This allows for a hands-on evaluation of the module's performance and the synchronization between the extracted timestamps and the actual video content.



Figure ‑ Video player showing first step

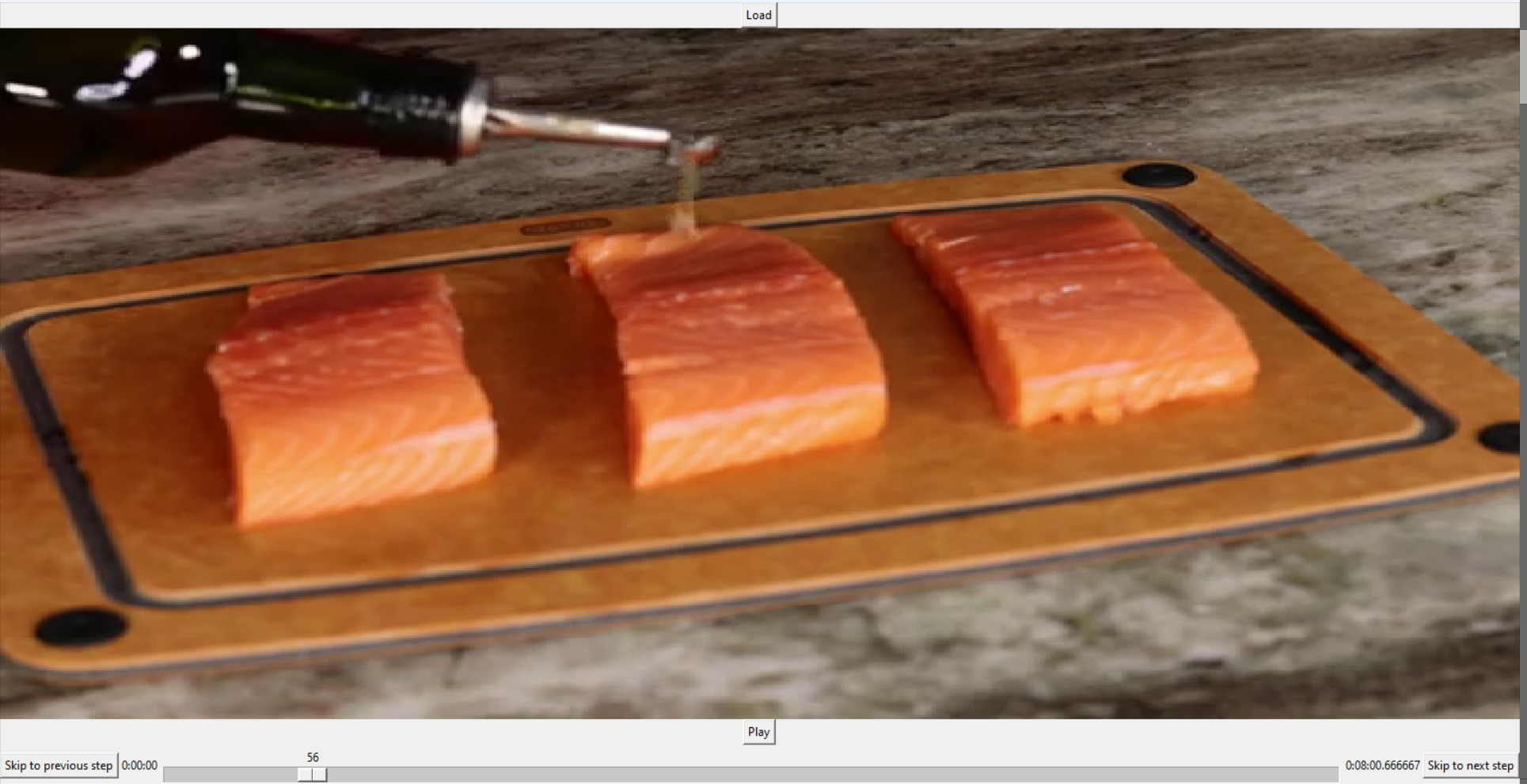
By utilizing the video player within our testing application, we were able to navigate through the video content in a structured manner. Each jump from one step to another corresponded to transitioning between different processes or segments within the video.

Figure ‑ Video player showing the next step

This structured navigation provided us with the ability to control and browse through the video, allowing us to conveniently move between distinct sections or steps. By leveraging the video player's controls, such as skipping to the next or previous steps, we were able to precisely examine and assess the content of each process within the video.

This approach facilitated a comprehensive understanding of the video's structure and allowed us to verify the accuracy and coherence of the step timestamps generated by the video analysis module.

### 5.2.2. Integration Testing

After conducting separate testing for each module, we proceeded to integrate the models into a demo application. The design of the modules facilitated the integration process, enabling us to seamlessly connect the different processes.

The chatbot, which accepts user input in text form, generates a query that is passed to the recommender system. The recommender system responds with recipe IDs, which are used to fetch the corresponding recipes. The chatbot can then present these recipes to the user.

This user-system interaction takes place through an API. The user submits their message via a POST request and receives a response from the chatbot, which includes the reply as well as the attached recipe.

Upon fulfilling the user's request, the recipe ID is forwarded to the video analysis module, which initiates its processing. The resulting output, an array containing the timestamps of each step, is sent back to the user's client. The video player utilizes these timestamps for navigation.

The clearly defined input and output formats for each module simplified the integration process. By ensuring consistent formatting, combining all the components together became more straightforward.

To thoroughly test the integrated application, we executed multiple runs of the main flow using various requests and recipes. This iterative testing approach allowed us to identify and address any issues or discrepancies, ensuring the smooth functioning of the integrated system.

## 5.3. Testing Schedule

Table ‑ Testing Schedule

|  |  |  |
| --- | --- | --- |
| Module | Start of testing | End of testing |
| Recommender | 27-4-2023 | 29-5-2023 |
| Chatbot | 7-5-2023 | 29-5-2023 |
| Video Analysis | 24-5-2023 | 30-5-2023 |
| Integrated Application | 31-5-2023 | 10-6-2023 |

## 5.4. Comparative Results to Previous Work

We couldn’t find a working application similar to ours to compare results produced, as most AI cooking applications generate recipes tailored to the user’s needs.

The closest application is Tasty chatbot a chatbot for recipe recommendations.

We can see that we produce results that satisfy the same requirements of Tasty.

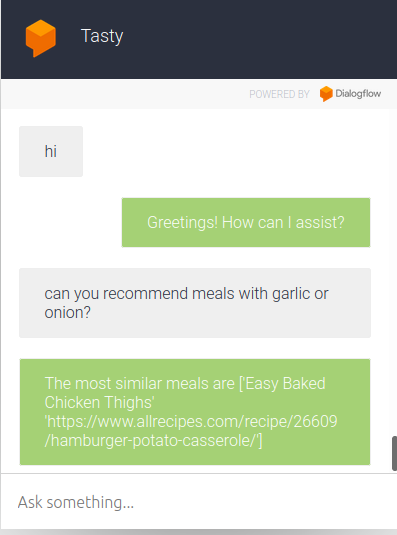


Figure ‑ Tasty vs Chef&1/2 Chatbot

We also have access to the testing results of the original paper for DSGR[10], which we can use to compare our model against the original model.

Table ‑ DSGR' original vs own

|  |  |  |
| --- | --- | --- |
| Dataset | NDCG@10 | Hit@10 |
| Beauty | **35.90** | **52.40** |
| Games | **55.70** | **75.57** |
| CDs | **51.22** | **72.43** |
| \*Recipes (Own) | **29.58** | **47.47** |

**\***Trained on own implementation of DSGR.

The table presented demonstrates the evaluation results for three datasets used in the paper, alongside our own dataset comprised of user recipes interactions. Although there is still potential for further improvement, we were satisfied to achieve results that closely align with the official evaluations of the models.

These evaluation results indicate that our dataset, derived from real user interactions with recipes, exhibits comparable performance to the datasets utilized in the paper. The close alignment between our results and the official evaluations suggests that our dataset accurately reflects user preferences and interactions, enabling our models to perform effectively in recommending recipes.

While there is room for enhancement, these promising outcomes validate the reliability and applicability of our models in the context of real-world recipe recommendation scenarios. The results serve as a positive indication of the effectiveness of our approach and highlight the potential for future improvements and optimizations in order to further enhance the performance of our recommendation system.

# Conclusions and Future Work

In this chapter we will be summarizing our project, the results we have found and the project’s features and limitations. We will also shed a light on the challenges we faced, and the experience we gained. We will also be discussing plans for future work for the product to be used commercially among people

## 6.1. Faced Challenges

In this part we will talk about the challenges we faced during the implementation of the project in every module.

### 6.1.1 Dataset

### 6.1.1.1 Recipes dataset

In order to address the lack of Arabic recipe datasets available, we took the initiative to create our own dataset. Given that existing recipe datasets were primarily in English, we recognized the need to scrape various websites to gather Arabic recipes along with their corresponding ingredients. This process involved extracting recipe information, such as ingredients and instructions, from multiple Arabic websites. By collecting and compiling this data, we were able to curate a comprehensive dataset specifically tailored to Arabic recipes. This resource will serve as a valuable asset for future research and development in the field of Arabic cuisine and culinary exploration**.**

### 6.1.1.2 chatbot dataset

In order to develop a chatbot capable of providing users with recipe recommendations and responding to their inquiries, we encountered the challenge of not having an appropriate dataset available. To overcome this obstacle, we undertook the task of creating our own dataset. We designed a data collection process that focused on understanding the intent of users and capturing their preferences regarding recipes. By engaging with users and analyzing their questions, requests, and feedback, we were able to gather valuable information that enabled us to build a dataset specific to recipe-related interactions. This dataset now serves as a crucial resource for training our chatbot, allowing it to accurately interpret user intent and provide relevant and personalized responses, ultimately enhancing the user experience and satisfaction.

### 6.1.2 Scene detection

In our pursuit of developing a scene detection model, we dedicated considerable effort to creating a comprehensive model that we believed would meet some requirements we wanted. However, despite our best intentions, the model ultimately fell short of our expectations and did not perform as efficiently as anticipated. We encountered challenges and limitations that prevented the model from delivering the desired results. Despite our initial assumptions and careful design, the model failed to meet our expectations as after implementing it, it was not able to detect the changes between 2 scenes.

### 6.1.3 Bridging the Gap Between Desired Features and Implementation

At the outset of our project, we had a clear vision of the features we aimed to develop. However, we soon realized that translating these aspirations into concrete implementations was a complex and daunting task.

Despite meticulous planning and research, we encountered unexpected technical limitations, resource constraints, and unanticipated complexities.

This process of discovery and adaptation pushed us to refine our understanding, improve our strategies, and collaborate to find practical ways to bridge the divide between our desired features and the practical realities of their implementation. Through dedication, resilience, and a commitment to problem-solving, we gradually unraveled the challenges, paving the way for the successful realization of our project objectives.

## 6.2. Gained Experience

In this part we will discuss what we learnt and experience gain during our journey in this project.

### 6.2.1 Enhancing search skills

Our project enhanced our search skills by providing us with practical experience and exposure to various search techniques, effectively filter information, and retrieve more relevant results, leading to improved research outcomes.

### 6.2.2 learned how to fine tune bert-model

We used a model called arabert which is a model trained to large Arabic data sets and we had to fine tune it to adapt it to a specific task or domain. The process entails retraining the model using a smaller, task-specific dataset, allowing it to learn more nuanced patterns and improve performance. Fine-tuning helps leverage the knowledge from the pre-trained model while tailoring it to a specific Arabic language application.

### 6.2.3 Teamwork and Planning

The project provided valuable experience in teamwork and planning by requiring us to collaborate effectively, allocate tasks, set deadlines, and communicate regularly to ensure smooth progress. We learned to adapt and coordinate efforts, leverage individual strengths, and navigate challenges together, fostering stronger teamwork and honing our planning skills for future projects.

## 6.3. Conclusions

In conclusion, our graduation project has successfully created a user-friendly application that enhances the cooking experience. By offering a good collection of recipes, detailed ingredient lists, and accompanied with instructional videos, we have empowered users to explore and expand their cooking knowledge. Throughout the development process, we gained valuable insights about our user needs also we had a great effective teamwork through working on this project. This project not only highlights our technical proficiency but also shows our dedication to make it easier on people's lives by making cooking more accessible and enjoyable. Our application has the potential to revolutionize how people approach cooking, inspiring creativity and culinary exploration for all.

## 6.4. Future Work

Future work for this project involves expanding the dataset to improve the chatbot's performance and personalization. Incorporating natural language processing techniques like sentiment analysis can help the chatbot understand user feedback better. Implementing a recommendation system based on user ratings and preferences would enhance the user experience. Integrating the chatbot with various platforms and devices, such as social media and smart home devices, would increase its accessibility. Continuous evaluation, refinement, and user testing are crucial to adapt the chatbot to evolving user needs also implementing a sophisticated video analysis model that identifies cooking actions and produces captions to annotate recipe videos.

**References**

[1] “Global food trends 2021: How our habits have changed, as told by social images | YouScan.” https://youscan.io/blog/food-trends/ (accessed May 23, 2023).

[2] J. Cooper, “Cooking trends: The Digital Kitchen - Think with Google,” 2015.

[3] “basyl - your AI cooking assistant.” https://www.basyl.co/create/basic (accessed May 27, 2023).

[4] “khadija267/Recipes-Recommendation-Chatbot: A project about recommeing recipes and ingredients based on the needs of the user.” https://github.com/khadija267/Recipes-Recommendation-Chatbot (accessed May 27, 2023).

[5] “What is a Recommendation System? | Data Science | NVIDIA Glossary.” https://www.nvidia.com/en-us/glossary/data-science/recommendation-system/ (accessed Jun. 01, 2023).

[6] “What is a Recommendation System? | Data Science | NVIDIA Glossary.” https://www.nvidia.com/en-us/glossary/data-science/recommendation-system/ (accessed May 27, 2023).

[7] A. Vaswani *et al.*, “Attention Is All You Need,” Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.03762

[8] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” [Online]. Available: https://github.com/tensorflow/tensor2tensor

[9] T. Wang, Y. M. Brovman, and S. Madhvanath, “Personalized Embedding-based Recommendations at eBay,” in *IEEE International Conference on Program Comprehension*, IEEE Computer Society, 2022, pp. 36–47. doi: 10.1145/nnnnnnn.nnnnnnn.

[10] M. Zhang, S. Wu, X. Yu, Q. Liu, and L. Wang, “Dynamic Graph Neural Networks for Sequential Recommendation,” Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.07368

[11] E. Guo *et al.*, “Learning to Measure Change: Fully Convolutional Siamese Metric Networks for Scene Change Detection,” Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.09111

[12] S. Benhur, “A friendly introduction to Siamese Networks | by Sean Benhur | Towards Data Science,” 2020. https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942 (accessed Jun. 02, 2023).

[13] H. Zhao, W. J. Wang, T. Wang, Z. Bin Chang, and X. Y. Zeng, “Key-frame extraction based on HSV histogram and adaptive clustering,” *Math Probl Eng*, vol. 2019, 2019, doi: 10.1155/2019/5217961.

[14] “My chef chatbot.” https://docs.google.com/forms/d/1fehKcQbOMr-Vk8RNq1Oubl5ztFF1iaxLYlPgSF1Mdes/viewform?ts=644befe6&edit\_requested=true (accessed Jun. 02, 2023).

[15] “What is named entity recognition (NER) and how can I use it? | by Christopher Marshall | super.AI | Medium.” https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d (accessed Jun. 03, 2023).

[16] “Word Embedding and Vector Space Models | by Jiaqi (Karen) Fang | Analytics Vidhya | Medium.” https://medium.com/analytics-vidhya/word-embedding-and-vector-space-models-11c9b76f58e (accessed Jun. 03, 2023).

[17] “Dimensionality Reduction: Singular Value Decomposition | by Antriksh Singh | Medium.” https://medium.com/@antriksh\_66433/dimensionality-reduction-singular-value-decomposition-727426d3b063 (accessed Jun. 03, 2023).

[18] “Demystifying NDCG. How to best use this important metric… | by Aparna Dhinakaran | Towards Data Science.” https://towardsdatascience.com/demystifying-ndcg-bee3be58cfe0 (accessed Jun. 04, 2023).

**Appendix A: Development Platforms**

**and Tools**

This appendix explains used tools, platforms, and hardware kits. Any ready-made module should be mentioned and discussed in this appendix. The appendix is divided into two main sections; one for the hardware and the other is for software. Within each section, you could add as much subsections as needed, according to the number of tools and platforms that you use in your project.

In this space, before the first section, write an introductory paragraph to the appendix

**A.1. Hardware Platforms**

A description of any used hardware platforms/kit should be written in this section. Each platform/kit is better described in a separate subsection. (A1.1..)

**A.2. Software Tools**

A description of any used software tool/package should be written in this section. Each tool/package is better described in a separate subsection (A2.1,..)

**Appendix B: Use Cases**

Include all your use cases

**Appendix C: User Guide**

Prepare a user guide for your project. Ensure that the guide is clear, detailed and easy for an ordinary customer to use your project. Employ figures and charts as needed to facilitate the use of your guide

**Appendix D: Code Documentation**

Your code or parts of the code you feel necessary could be included here (optional) however for one copy of this report an attached CD with all of the code is a must.

Remember you will deliver three copies of this report.

**Appendix D: Feasibility Study**

Give a detailed feasibility study of your project