Faculdade de Engenharia da Universidade do Porto

Chatbot for food preferences modelling and recipe recommendation

Álvaro Miguel Figueira Mendes Samagaio

Dissertation

Mestrado Integrado em Bioengenharia - Engenharia Biomédica

Supervisor: Henrique Lopes Cardoso, PhD, FEUP | LIACC

Co-supervisor: David Ribeiro, MSc, Frauhofer Portugal

June 2020

Chatbot for food preferences modelling and recipe recommendation

Álvaro Miguel Figueira Mendes Samagaio

Mestrado Integrado em Bioengenharia - Engenharia Biomédica

June 2020

Abstract

Malnutrition is one of the leading causes of death in the world. It leads to the development of several conditions that negatively affect the life quality of the populations and it represents a major burden to the healthcare systems and society. In fact there are an estimated 3.15 billion people that suffer from some form of malnutrition, being 2.69 billion of those overweight or obese. Moreover, the estimated worldwide economical impact of malnutrition and related conditions amounts to $3.5 trillion. There is a close relation between poor dietary habits and malnutrition. This relation aggravates in the elderly population where a healthy diet takes a key role in an active and healthy ageing, allowing for a longer independent life and improved life quality. Therefore, there is an obvious need for dietary counselling and supervising. The appearance of nutritional companion mobile applications that enable the access to information and easy supervision, often integrating activity monitors, tries to tackle the aforementioned need. More often than not, the interaction between the system and the user happens through on-screen information and touch-based inputs. This form of interaction can sometimes be an obstacle to the regular adoption of these technologies, especially in the elderly population.

Accordingly, this document reports the work done in a Master's Thesis developed at *Fraunhofer Portugal*, a non-profit research institute that has been recently involved in the development of active ageing technologies. Researchers at *Fraunhofer* have developed a recommender system (Lifana) that creates meal plans for each week and also allows for the access to nutritional information of meals and ingredients. The main goal of the thesis is the development of a conversational agent that should be capable of replacing the screen-based interaction between the users and the recommendation engine, thus providing a more immersive and natural experience.

Taking this into account, the work focuses on the construction of a chatbot that is able to recognise and classify the user intention in a conversation turn and take an action based on this. This corresponds to two main parts of the conversational agent: the Natural Language Understanding and the Dialogue Management. Both parts required the collection of a proper dataset that was further annotated and stored. Regarding Natural Language Understanding, it was divided in two tasks: intent classification and entity extraction, which enables parsing sentences into structured information, that can be processed. An extensive experimentation protocol was followed mixing several feature sets and classification models. The approach that provided the best results employs a transformer-based architecture and performs joint intent classification and entity recognition. This joint approach reached a F1 score of 0.91 for entity recognition and 0.88 for intent classification.

A set of experiments was also conducted for Dialogue Management, where once again the transformer based architecture showed the best performance in action prediction. This model reached an accuracy of 0.82. Moreover, this document also describes the way the several actions were encoded in to the chatbot, so that communication between the user and the Lifana recommender system is possible.

Two additional components that take very important roles in the proper functioning of the chatbot were developed: the preferences model and the food retrieval algorithm. The former uses VADER Sentiment Analysis tool to extract sentiment polarity and intensity and, based on that, create a suitable rating mechanism for Lifana. This allows for preference expression using

Natural Language and hence qualitative terms, instead of a numeric rating, improving the user experience of the chatbot. The final algorithm produced an overall Mean Absolute Error of 0.86 in a 0 to 5 range, when compared to the data collected for this purpose (sentiment intensity). Regarding the food retrieval algorithm, it works by extracting the most correct entries from a food terms database using a matching mechanism based on word embeddings similarity. These embedding were created by retrofitting ConceptNet Numberbatch pre-trained embeddings with semantic information from LanguaL, a food ontology. The retrofitting was improved by the utilisation of TF-IDF weighting. The results for ingredient classification showed a high improvement when comparing the retrofitted embeddings with the non-retrofitted embeddings. This algorithm was validated through user opinion

The developed chatbot was evaluated according to user testing. For this purpose a usability test based on tasks that mimic the most common use cases of Lifana was designed. Users were able to complete the tasks and after evaluate the bot according to three different usability questionnaires the *System Usability Score*, the *User Experience Questionnaire* and the *Chatbot Usability Questionnaire*. The chatbot scored above all benchmarks, showing that the actions were correctly implemented and articulated. Also, a comparative analysis of the questionnaires and metrics was performed to assess the validity and the overlap of the evaluation. Additionally, the evaluation of the food retrieval algorithm showed that user prefer the semantic-based method (word embeddings) over the word-matching search algorithm (77% vs 23%). The former proved to be more precise than the latter.

All the development stages were performed using Rasa framework.

These results proved the feasibility of this work even though there is still space for further developments and testing. The chatbot is expected to keep improving in performance with the collection of more data.

Resumo

A malnutrição é uma das principais causas de morte no mundo, conduzindo ao desenvolvimento de várias condições que afectam negativamente a qualidade de vida das populações. Para além disto, representa um fardo considerável para os sistemas de saúde e para a sociedade. De facto, estima-se que haja 3,15 mil milhões de pessoas que sofrem de alguma forma de malnutrição, sendo 2,69 mil milhões desses obesos ou com peso excessivo. Adicionalmente, o impacto económico mundial estimado da malnutrição e condições relacionadas ascende a 3,5 biliões de dólares. Existe uma relação estreita entre os maus hábitos alimentares e a malnutrição. Esta relação agrava-se na população idosa, onde uma dieta equilibrada assume um papel fundamental para um envelhecimento ativo e saudável, permitindo uma vida independente mais longa e uma melhor qualidade de vida. Por conseguinte, existe uma necessidade óbvia de aconselhamento e supervisão dietética. O aparecimento de aplicações móveis de acompanhamento nutricional que permitem o acesso à informação e a fácil supervisão, muitas vezes integrando monitores de actividade, tenta dar resposta à referida necessidade. Na maior parte das vezes, a interacção entre o sistema e o utilizador acontece através de informação no ecrã e inputs baseados no toque. Esta forma de interação pode, por vezes, ser um obstáculo à adoção regular destas tecnologias, especialmente na população idosa.

Consequentemente, este documento relata o trabalho realizado numa Tese de Mestrado desenvolvida na *Fraunhofer Portugal*, um instituto de investigação sem fins lucrativos que esteve recentemente envolvido no desenvolvimento de tecnologias de envelhecimento activo. Investigadores da *Fraunhofer* desenvolveram um sistema de recomendação (Lifana) que cria planos de refeições semanais e também permite o acesso à informação nutricional de refeições e ingredientes. O principal objectivo desta dissertação é o desenvolvimento de um agente de conversação que deverá ser capaz de substituir a interação baseada no ecrã entre os utilizadores e o motor de recomendação, proporcionando assim uma experiência mais imersiva e natural, através da conversa.

Tendo isto em conta, o trabalho centra-se na construção de um chatbot que seja capaz de reconhecer e classificar a intenção do utilizador a cada turno da conversa e tomar uma ação com base nisso. Para tal, o sistema pode ser dividido em duas partes principais: o módulo de Compreensão de Linguagem Natural e o módulo de Gestão do Diálogo. Ambas as partes exigiram a recolha de dois conjunto de dados, que foram devidamente anotados e armazenados. Estes conjuntos de dados foram utilizados para treinar e avaliar os classificadores implementados durante o desenvolvimento.

Relativamente ao módulo de Compreensão de Linguagem Natural, este foi dividido em duas tarefas: classificação da intenção e reconhecimento de entidades mencionadas, o que permite transformar a informação não estruturada presente no texto em informação estruturada, que pode ser processada pelo sistema. De modo a escolher a melhor abordagem, uma série de experiências foram conduzidas, combinando vários conjuntos de características e modelos de classificação. A abordagem que proporcionou os melhores resultados emprega uma arquitectura baseada em *transformers* e realiza conjuntamente a classificação da intenção e o reconhecimento de entidades. Para além disto, esta abordagem utiliza também vetores de palavras pré-treinados provenientes do GloVe. Esta abordagem conjunta atingiu um valor F1 de 0,91 para o reconhecimento de entidades e de 0,88 para a classificação de intenções.

Foi também realizado um conjunto de experiências para o módulo de Gestão do Diálogo onde, mais uma vez, a arquitectura baseada em *transformers* mostrou o melhor desempenho na previsão da ação. Este modelo atingiu uma exatidão de 0,82. Além disso, este documento descreve também a forma como as várias ações que mimetizam os casos de uso do sistema de recomendação foram codificadas no chatbot, para que seja possível a comunicação entre o utilizador e o sistema de recomendação Lifana.

Foram desenvolvidos dois componentes adicionais que assumem papéis muito importantes no bom funcionamento do chatbot: o modelo de preferências e o algoritmo de extração de alimentos da base de dados. O primeiro utiliza a ferramenta VADER, que realiza análise de sentimento em texto, para extrair a polaridade e intensidade dos sentimentos e, com base nisso, criar um mecanismo de classificação adequado para o sistema Lifana. Isto permite a expressão de preferências utilizando Linguagem Natural e, portanto, termos qualitativos, em vez de uma classificação numérica, melhorando a experiência do utilizador. O algoritmo final produziu um erro geral médio absoluto de 0,86 num intervalo de 0 a 5, quando comparado com os dados recolhidos para este fim (intensidade do sentimento). Foi também necessário adaptar o algoritmo de modo a considerar os extremos do espectro, pois estes tem uma importância acrescida no sistema Lifana. Relativamente ao algoritmo de extração de alimentos, este funciona extraindo as entradas mais corretas de uma base de dados de termos alimentares (ingredientes e receitas) utilizando um mecanismo de correspondência baseado na semelhança nos vetores de palavras. Estes vetores foram criados através da adaptação de vetores pré-treinados ConceptNet Numberbatch com a incorporação da informação semântica do LanguaL, uma ontologia alimentar. Para tal, foi utilizada a técnica de *retrofitting*. O retrofitting foi melhorado através da utilização de uma ponderação baseada em TF-IDF. Os resultados da classificação dos ingredientes mostraram uma grande melhoria ao comparar os vetores que sofreram retrofit com os vetores que não sofreram o processo de *retrofitting*. Este algoritmo foi validado através da opinião do utilizador.

Para além das métricas referidas anteriormente, o chatbot desenvolvido foi avaliado de acordo com testes feitos por utilizadores. Para este efeito, foi concebido um teste de usabilidade baseado em tarefas que mimetizam os casos de uso mais comum de Lifana. Os utilizadores teriam de completar as tarefas e depois avaliar o bot de acordo com três questionários de usabilidade diferentes o *System Usability Score*, o *User Experience Questionnaire* e o *Chatbot Usability Questionnaire*. O chatbot pontuou acima de todos os pontos de referência para os vários questionários, mostrando que as ações foram corretamente implementadas e articuladas. Além disso, uma análise comparativa dos resultados dos questionários foi também efetuada, com o objetivo de perceber se estes avaliavam os mesmos fatores no chatbot. Adicionalmente, a avaliação do algoritmo the extração de alimentos mostrou que os utilizadores preferem (77% vs 23%) o método baseado na semântica (vetores de palavras) que o método que procura por correspondência literal das palavras, sendo que o pimeiro é mais preciso que o segundo.

Todo o desenvolvimento do chatbot foi realizado recorrendo à ferramenta Rasa.

Os resultados obtidos neste trabalho demonstram a exequibilidade do desenvolvimento de um chatbot como meio de interação de um sistema de recomendação. No entanto, existe ainda espaço para melhorias relativamente ao desenvolvimento e à testagem. Será expectável que o chatbot continue a melhorar a sua prestação com a recolha de mais dados.

Agradecimentos

Esta tese, resumida neste documento, representa o culminar de todos os meus anos de universidade, desde trabalho a boémia com muitas memórias. Todo este tempo é dedicado especialmente ao meu Avô Álvaro que por um infortúnio do destino partiu a metade do percurso e não teve a oportunidade de ver o seu neto de bengala e cartola, coisa que lhe traria das maiores felicidades da vida.

Espero ter honrado o teu nome e que te tenha trazido orgulho com a minha vida. Este trabalho é teu Vuvu. Um eterno Obrigado

Também quero fazer referência ao meu avô António que não tive oportunidade de conhecer tão bem mas espero que esteja também orgulhoso do neto e da filha que o criou.

Para não destoar do tom, e como bom Português, começo os agradecimentos com um sentimento saudosista por aquilo que nos foi roubado este ano. Tudo aquilo que não pudemos viver neste último período da vida de estudante. Todos encontros e todas as celebrações da vida e da liberdade de ser um estudante na cidade do Porto. Não é justo, não foi justo. Por quê a nós? Não podia ter sido o semestre passado? ou o próximo? O ano de finalista tem um sabor bastante agridoce do início ao fim. Acabados de chegar de Erasmus, um semestre onde muitos de nós experiencia pela primeira vez a liberdade total e onde a maior parte das coisas tem uma leveza ináta que é ótima de usufruir. O primeiro semestre começa com a sensação de que somos os maiores da FEUP. No entanto, rapidamente isso é atropelado por uma quantidade absurda de trabalho que nos é pedido, a par da tese, o semestre mais trabalhoso sem ponta de dúvida. Entramos, então, no segundo semestre com uma sensação de que agora é que vai ser, O semestre! Vamos poder fazer tudo o que não fizemos porque nunca tivemos tempo, ou simplesmente porque nunca esteve a acabar o tempo. Se iria ser o semestre ninguém saberá. Todos os planos adiados ou cancelados. Todas as ideias proteladas para outra altura e vimo-nos obrigados a ficar em casa durante todo o último tempinho de vida de estudante que muitos terão. Até a Queima e as suas celebrações nos roubaram. Quem nunca se imaginou naquele palco a rasgar um professor de alto a baixo, de benagala na mão e cartola na cabeça, quem nunca sentiu a felicidade do cortejo onde tudo é bom e feliz, nunca viveu uma Queima na sua plenitude. Nem o final do curso escapou, este ano os finalistas acabam o curso a desligar uma chamada de Zoom depois da defesa virtual. Ninguém tinha em mente um final tão aborrecido para um marco tão importante na vida. O final do que dizem ser os melhores anos, que não podemos desperdiçar.

Soube a pouco

No entanto, vamos para sempre ser o Ano da Pandemia, a infame *Class of 2020*. Olhando para trás sabemos que vai tudo deixar saudade, desde as noitadas nas salas de estudo do B, aos cafézinhos a seguir ao almoço no Bar de Minas ou na ESE, passando pelas noites nos mágicos FEUPCaffes e os finos na AE. Não podia deixar de referir Metal&Bio que me proporcionou bastantes noites de boémia académica e cujo caminho se separou do meu nos últimos anos do curso.

Quero agradecer também a todos os amigos que fizeram este percurso comigo ou que de alguma maneira estiveram presentes nele. Começando pelo grupo da faculdade, um grande obrigado aos amigos de Biomédica que desde o terceiro ano se foram aproximando cada vez

mais e nunca pensei que fosse possível fazer tantos planos com um grupo tão grande. Isso e boatos, boatos sempre. Obrigado às gémeas Raquel e Leonor, que apesar de não achar que sejam parecidas, aceito a designação. Os vossos resumos do grupo de cadeiras desiganadas trivialmente como biomecânica foram cruciais no meu desempenho académico. Obrigado Raquel por te rires incondicionalmente das minhas piadas geniais. Obrigado aos Pedros como uma entidade conjunta feita por Pedro Dias aka Pedrito, Pedro Ribeiro aka Pedrão e Miguel Meneses aka Migas. Pedro Dias, o único gajo mais stressado que eu com o que os outros já sabem. Pedro Ribeiro como o único outro membro do grupo que me pode fazer frente em termos de tamanho, gigante, e o gajo com menos paciência para as minhas cenas. Obrigado Migas pela companhia nas boleias e na Fraunhofer, incluindo lanches e almoços. Obrigado gémeos Bruneses, Bruno Santos e João Meneses. Bruno, o gajo dos esquemas que têm tudo para correr mal mas que no fim até correm bem. Obrigado pela tua insistência e amizade, mesmo sendo o membro mais recente no grupo e provavelmente o que leva com mais negas. Jonas, um grande obrigado também pela tua disponibilidade quase total ao longo deste tempo e por nunca deixares o grupo morrer. Tenho de referir também o núcleo duro que me acompanhou desde os primeiros dias e nas viagens que fizemos ao longo do curso. Obrigado Nataliya aka Nata, a pessoa mais calma, impávida, serena e com as vibes mais retro que conheço. Obrigado Joana aka Jupi, apesar de andarmos um bocado às turras sei que somos bons amigos. Ana aka Anokas, a pessoa que mais discorda de mim, com as ideias mais opostas às minhas em literalmente tudo, nem me lembro de um dia concordarmos em algum tópico. Estas discussões eternas têm de continuar a acontecer. Obrigado por me mostrares o outro lado e que às vezes o ego chateia um bocado. Diogo aka Gu(Gu) aka Vitória aka Gustavo aka uma infinidade de outros nomes derivados desde o início que estás no topo da lista e és O amigo da faculdade. Ana e Gu, obrigado por me aturarem estes anos todos e ainda serem meus amigos. Por me ajudarem nas alturas mais chatas e mais infelizes e por partilharem comigo os melhores momentos. Sei que muito do que consegui fazer na faculdade tem mão vossa e sem um núcleo destes torna-se tudo muito mais difícil e menos agradável. Da faculdade só falta o Diogo Malafaya aka Malafaya, que andou desaparecido durante dois anos mas depois voltou para ser o parceiro de inúmeros projetos inacabados e de escrita deste documento. O gajo das ideias que nunca atende as chamadas mas que continua a ser um bom amigo. Um especial agradecimento à segunda casa, o mítico e lindo 3ºD da rua Padre (professor?) António Cruz. Dizem que os amigos da faculdade são para a vida, espero que sejam!

Seguindo nos amigos, um especial agradecimento ao pessoal do basket, que paralelamente à faculdade foi sendo outro grupo de amigos muito próximo e que certamente vai ficar comigo até um dia sermos todos magnatas. O companheirismo é forte neste. Obrigado Rodrigo, Vítor, Bernardo, Miguel, André, João Maria, Branco e até o desaparecido do Zé.

Agradeço a todas as outras pessoas que se cruzaram comigo nestes anos, nomeadamente ao pessoal de 96 que fez comigo as primeiras cadeiras do MIB (obrigado Paulo Maia e Cris por vos andar a chatear estes anos todos), ao pessoal da fantasy da NBA e que esta ocorra todos os anos, ao pessoal de mecânica que frequenta o mítico 3º Direito (incluindo o Natal) e geriam a melhor barraca da Queima do Porto, a Dartacão, ao pessoal da ShARE que já lá vai há uns anos mas ficaram marcados, às amigas da Nocas, principalmente à Carol que fez de casamenteira e, por fim, ao pessoal dos Cedros que, não estando tão presentes durante a faculdade, são sempre um grupo importante no meu percurso.

Um também muito especial agradecimento à minha Nocas que foi a minha companhia durante mais de metade do curso e que passou e aturou vários momentos felizes e infelizes do percurso. Sempre a apoiar as minhas ideias sem nexo e as pancas. Sei que sou uma pessoa melhor contigo e tiveste um papel fundamental na minha vida académica para além da vida corrente. Um apoio incondicional todos os dias e todos os momentos. Fazemos, de facto, uma a melhor das equipas. A cada dia que passa ficamos mais próximos e com o futuro mais sólido. E obrigado aos pais pelos jantares à pressa antes dos treinos e por criarem a melhor pessoa com quem me cruzei nos últimos anos.

Não podia deixar de referir a minha família que foi a base e o sustento que me permitiu

chegar onde cheguei e concretizar os planos que imaginei. Aos meus pais que sempre me suportaram e me motivaram a ser o melhor naquilo que fazia, sendo a figura exemplar que eu acho que são, muito obrigado. Aos meus avós que me criaram desde pequenino e permitiram que os meus pais me pudessem sustentar e aos meus estudos e atividades para enriquecer a minha formação, muito obrigado. Ao meu irmão que é o meu eterno companheiro da vida, também um obrigado. Espero poder ser um exemplo para ele e gostava de dizer-lhe para aproveitar cada segundo desta vida de estudante, ainda por cima agora que já estiquei a corda aos pais. Aproveita Kiko enquanto ela está solta.

Um especial obrigado aos docentes e técnicos que me acompanharam neste percurso, nomeadamente ao Professor Henrique que me orientou na tese e ao David que me supervisionou da parte da Fraunhofer. Sem os vossos valiosos inputs o trabalho não teria metade da qualidade.

Encerro estes agradecimentos com um sentimento de dever cumprido e orgulho. Estou pronto para a próxima etapa que aí vem. Espero que esteja ao nível desta que passou porque deixou as expectativas bem altas. É tão bom ser estudante em Portugal. Mas o compromisso é o mesmo, ser o melhor que puder.

Strive for Greatness
- LeBron Raymone James

Contents

List of Figures xiii

List of Tables xvii

List of Abbreviations xix

1 Introduction 1
- 1.1 Motivation . . . 4
- 1.2 Goals . . . 6
- 1.3 Contributions . . . 7
- 1.4 Document Structure . . . 8

2 Background 11
- 2.1 Malnutrition . . . 11
  - 2.1.1 Malnutrition and Diet . . . 13
  - 2.1.2 Healthy diet . . . 13
- 2.2 Lifana Nutrition Recommender System . . . 15
  - 2.2.1 Cloud server component . . . 15
  - 2.2.2 Knowledge bases description . . . 19
- 2.3 Natural Language Processing . . . 21
  - 2.3.1 Approaches to Natural Language Processing . . . 21
  - 2.3.2 Natural Language Processing tasks . . . 22
  - 2.3.3 Natural Language Processing based on embeddings . . . 25
- 2.4 Summary . . . 27

3 State of the Art 29
- 3.1 Chatbots and Communication . . . 29
  - 3.1.1 Speech Recognition and Synthesis . . . 31
  - 3.1.2 Communication as an action . . . 36
  - 3.1.3 Intent Determination . . . 37
  - 3.1.4 Dialog Management and Response Generation . . . 38
  - 3.1.5 Knowledge bases . . . 40

3.2 Chatbot construction tools . . . 41
3.2.1 Speech Recognition and Speech Synthesis . . . 41
3.2.2 Intent Identification and Dialogue Management frameworks . . . 43
3.3 Chatbot evaluation . . . 50
3.4 Nutrition and Health Related Chatbots . . . 51
3.5 Summary . . . 53

4 A Chatbot for Lifana **55**
4.1 Problem description . . . 55
4.1.1 Lifana API . . . 56
4.2 Available data . . . 57
4.2.1 Food related data . . . 57
4.2.2 Intial NLP data . . . 58
4.3 Rasa Framework . . . 59
4.3.1 Rasa NLU . . . 59
4.3.2 Rasa Core . . . 63
4.3.3 Extra features . . . 68
4.4 Use Cases Specification . . . 69
4.4.1 Meal Plan Actions . . . 69
4.4.2 Nutritional Content Actions . . . 70
4.4.3 Food Preference Actions . . . 71
4.4.4 User Related Actions . . . 71
4.5 Summary . . . 72

5 Chatbot Development **73**
5.1 NLU approach . . . 73
5.1.1 NLU Data . . . 73
5.1.2 Techniques Description . . . 77
5.1.3 Intent Classification . . . 79
5.1.4 Entity Recognition . . . 83
5.1.5 Joint Intent Classification and Entity Extraction . . . 85
5.1.6 Regex Features . . . 86
5.1.7 Extra Components . . . 88
5.1.8 NLU Final Pipeline . . . 90
5.2 Preference Modelling . . . 92
5.2.1 VADER NLTK Sentiment Analyser . . . 93
5.2.2 Google Cloud Natural Language API . . . 94
5.2.3 Sentiment Analysis for preference modelling . . . 94
5.3 Food matching . . . 101
5.3.1 ConceptNet Numberbatch and Retrofitting . . . 102

5.3.2 Experimental Evaluation . . . 103
5.4 Dialogue Management . . . 110
5.4.1 Responses . . . 110
5.4.2 Dialogue Management Dataset . . . 115
5.4.3 Dialogue Management Experimental Evaluation . . . 116
5.5 Summary . . . 117

6 User Experience Evaluation **119**
6.1 Experimental Protocol – Usability Test . . . 119
6.1.1 Test Tasks . . . 120
6.1.2 System Usability Scale . . . 121
6.1.3 User Experience Questionnaire . . . 121
6.1.4 Chatbot Usability Questionnaire . . . 122
6.1.5 Food Matching Validation . . . 123
6.2 Results . . . 124
6.2.1 Single Ease Question . . . 124
6.2.2 Completion times . . . 126
6.2.3 System Usability Score . . . 126
6.2.4 Chatbot Usability Questionnaire . . . 128
6.2.5 Questionnaire Comparison . . . 129
6.2.6 Food matching . . . 131
6.3 Summary . . . 133

7 Conclusion and Future Work **135**

Bibliography **139**

A Complementary Information **155**

B Complementary Information **173**

List of Figures

1.1 Demographic projections for the population according to Eurostat [1]. The percentage of old people (age $\geq$ 65) and the Old Dependency Ratio for the EU countries and Portugal are repesented in this Figure 2

2.1 Lifana recommender system sequential stages [23] 17
2.2 Left: Lifana main screen; Centre: User profile; Right: Food Restrictions 18
2.3 App screens for meal plan, recipe information and nutritional content 18
2.4 Example of the food database used by the Lifana system 20
2.5 Named Entity Recognition example obtained using the StanfordNLP toolkit [80] 23
2.6 Penn Treebank Project [81] part-of-speech tagging [80] 23
2.7 Dependency parsing example. The tree structure represents the grammatical dependencies between the token of the sentence: *Bell, based in Los Angeles, makes and distributes electronic, computer and building products* [88] 25

3.1 Classification parameters of chatbots adapted from [95] 30
3.2 Usual chatter bot architecture. The system comprises several modules: speech recognition (speech-to-text), language understanding, dialogue manager, textual response generator and a text-to-speech synthesizer [80] 30
3.3 Comprehended steps in the Text-to-Speech process: Text analysis, prosodic and phonetic analysis and Speech synthesis [80] 34
3.4 Comparison between the DeepMind's Text-to-Speech approach (WaveNet) and other traditional approaches, regarding Human preference for both US English and Mandarin Chinese. The results were obtained using Mean Opinion Scores (from 1 to 5), which are the standard measure for subjective sound quality tests. The opinions came from a blind test in human subjects, from more than 500 ratings on 100 test sentences, adapted from [102]. 36
3.5 Dialogflow intent matching example [133] 44
3.6 Dialogflow intent matching and response flow [133] 45
3.7 Dialogflow follow-up intent example with a haircut appointment. The intents are combined by the input and output contexts [136]. 46
3.8 Interaction between a system and the Dialogflow API. It is processed according to the below described steps [137]. 47

4.1 Graphical representation of the scope of the dissertation work. The agent establishes the communication between the recommender system and the user. 56

4.2 Simplified schema of the database including only the main plan elements and their relationships . . . 58
4.3 Illustrating example of some samples for three different intents. Note that entities are annotated using "[]", with the correspondent entity name and synonym between round brackets . . . 60
4.4 Dual Intent and Entity Transformer model architecture. The model was developed by Bunk et al. and aims at giving a state-of-the-art performance while also maintaining a reduced inference time [171] . . . 62
4.5 An example of a story that shows the planned meals for a given time. It saves all the slots set at that moment as well as the intent and entities extracted by the NLU module and the bots responses. Marked with "*" are the user intents wheres the indented lines starting with "-" denote the bots actions or slots . . . 64
4.6 Transformer Embedding Dialogue Policy architecture and data processing along two conversation turns [179] . . . 66

5.1 Synonyms defined for the word "DINNER" . . . 90
5.2 Histogram of hits and misses according to model confidence for intent classification . . . 92
5.3 Distribution of samples per intensity class . . . 95
5.4 Values of Mean Absolute Error calculated per class for both models . . . 97
5.5 Difference in predictions for VADER's model before and after applying the capping rule described in Equation 5.2, for classes 0 and 5 . . . 98
5.6 Difference in predictions for Google Cloud's model before and after applying the capping rule described in Equation 5.2, for classes 0 and 5 . . . 99
5.7 The plots show the top 12 most frequent word lemma for each class. It is possible to see that the central classes (1 to 4) have increased variability. On the contrary, classes 0 and 5 are more heterogeneous, having some terms that appear much more than the others. . . . 100
5.8 Textual responses examples . . . 110

6.1 Volunteer chatbot use frequency . . . 124
6.2 Average values for Pre-task and post-task difficulty assessment question . . . 125
6.3 Average difference values between preconceived ease and real ease per task . . . 125
6.4 Average time to completion per task for the volunteers. The time taken by the developer is also represented by the line as a term of comparison . . . 126
6.5 Box-plot representing the SUS scores obtained in the test. The average is 84.7, the minimum was 67.5 and the maximum was 97.5. It is possible to observe that the first quartile is above the benchmark value 68.0 . . . 127
6.6 Benchmark comparison for each factor evaluated by UEQ . . . 128
6.7 Box-plot representing the CUQ scores obtained in the test. The average is 82.9, the minimum was 65.6 and the maximum was 98.44. . . . 129
6.8 SUS vs UEQ scores per volunteer plot . . . 129
6.9 SUS vs CUQ scores per volunteer plot . . . 130
6.10 UEQ vs CUQ scores per volunteer plot . . . 130
6.11 Response frequency for food matching precision assessment . . . 131

6.12 Response frequency for food matching recall assessment 132
6.13 Response frequency for food matching model preference 133

A.4.1 Confusion matrix obtained for intent classification using the final NLU pipeline depicted in Section 5.1.8. 172

B.1.1 Rasa X user interface used for user experience testing. This figure illustrates the initial message - before profile creation. It describes the scope of the chatbot and asks for personal detail collection 173
B.1.2 Rasa X user interface used for user experience testing. This figure illustrates the further steps of personal detail collection 174
B.1.3 Rasa X user interface used for user experience testing. This figure illustrates the last steps of profile creation 174
B.1.4 Rasa X user interface used for user experience testing. This figure illustrates the process of requesting to see a plan. Also, the welcome message is customised according to the user's name. 175
B.1.5 Rasa X user interface used for user experience testing. This figure illustrates the process of plan creation 175
B.1.6 Rasa X user interface used for user experience testing. This figure illustrates the ingredient consultation 176
B.1.7 Rasa X user interface used for user experience testing. This figure illustrates the process of requesting nutritional information 176

List of Tables

2.1 Body Mass Index reference values according to the World Health Organisation [47] 12
2.2 Minerals and vitamins and their daily nutrient reference values [61] 14
2.3 Daily reference intake values for macronutrients [61] 14
2.4 Number of created recipes for each meal time [23] 20
2.5 Example of stemming using the Porter algorithm [83] 24

3.1 Speech-to-Text tools comparison 43
3.2 Text-to-Speech tools comparison 43
3.3 Comparison between several available frameworks for chatbot creation – intent identification and dialog management 49

4.1 Delineated Use Cases to be integrated in the chatbot 72

5.1 Description of the entities used by the agent 76
5.2 Description of the defined intents and corresponding number of samples in the dataset 77
5.3 Sparse features extracted for entity recognition using CRF. The description is in Table 5.4 79
5.4 Description of the sparse features used for entity recognition 79
5.5 Pipelines tried for intent classification 81
5.6 Results for intent identification using the different configurations described above 82
5.7 Pipelines tried for entity recognition 84
5.8 Results for entity recognition using the different configurations described above . 85
5.9 Pipelines tried for joint intent classification and entity recognition 85
5.10 Results obtained for joint entity extraction and intent classification using DIET classifier 86
5.11 Pipelines tried for joint intent classification and entity recognition with Lookup table features 87
5.12 Entity extraction and intent classification results for the configurations using lookup tables as a way to improve performance 88
5.13 Individual performance scores for the entities affected by the addition of a lookup table (*food* and *person_name*) 88
5.14 Intent classification and entity recognition results for the final pipeline 91

5.15 Spearman and Person correlation, Mean Absolute Error and Root Mean Squared Error for the complete dataset, per sentiment analyser . . . 96
5.16 Measures of central tendency per class, for each classifier (both capped and not capped) . . . 96
5.17 Average values of MAE calculated by class for both models before and after capping 97
5.18 Details about ingredient data . . . 104
5.19 Accuracy results for ingredient classification according to *LanguaL* facets, using several different pre-trained embeddings models . . . 104
5.20 Accuracy results for ingredient classification according to *LanguaL* facets, using three different retrofitting approaches . . . 106
5.21 Accuracy results for ingredient classification according to *LanguaL* facets, using approach A for retrofitting with different methods of TF-IDF weighting . . . 108
5.22 Intent-Response mapping. The identification of any of those intents will automatically trigger the respective action as a response from the conversational agent . . . 116
5.23 Dialogue Management pipelines tried. The pipelines change in the used Policies for action prediction . . . 117
5.24 Dialogue Management approaches results . . . 117

6.1 Factors evaluated by the User Experience Questionnaire . . . 122
6.2 Chatbot Usability Questionnaire questions. . . . 123
6.3 t-Student pairwise test probability. The values are all below 0.05, meaning that the difference is statistically significant. . . . 126
6.4 UEQ results for each factor and comparison to the Benchmark. The average calculated by averaging each volunteer mean result is also presented (scale: [-3;3]) . 127
6.5 Pairwise Pearson correlation values for the several questionnaires used . . . 130
6.6 t-Student probabilities for pairwise combination of each questionnaire . . . 131

A.1 SVM parameters for grid search optimization. The underlined values are the ones used . . . 170
A.2 Parameters and Hyperparameters used in DIET Classifier . . . 170
A.3 Possible categories for each LanguaL facet . . . 171

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interfaces
APP	Application
ASR	Automatic Speech Recognition
BERT	Bidirectional Encoder Representations from Transformers
BMI	Body Mass Index
BOW	Bag-of-Words
CRF	Conditional Random Fields
CBOW	Continuous Bag-Of-Words
CUQ	Chatbot Usability Questionnaire
EU	European Union
FCDB	Food Composition Data Base
GMM	Gaussian Mixture Models
ICT	Information and Communications Technology
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
NCD	Non-Communicable Disease
NER	Named Entity Recognition
NHS	National Health System
NLP	Natural Language Processing
NLU	Natural Language Understanding
POS	Part-of-Speech
RDB	Relational Data Base
REGEX	Regular Expressions
RNN	Recurrent Neural Networks
SEQ2SEQ	Sequence to Sequence
SQL	Structured Query Language
SSML	Speech Synthesis Markup Language
STT	Speech-To-Text
SUS	System Usability Scale
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
TTS	Text-to-Speech
UK	United Kingdom
US	United States
UEQ	User Experience Questionnaire
WHO	World Health Organization

Chapter 1

Introduction

The world population is ageing rapidly. In 2030, the EU population aged over 65 is estimated to reach 24.1% of the total EU population. This number is expected to increase to 29.1% in 2060. In Portugal, as shown in Figure 1.1, this number is expected to increased from 22.1% in 2020 to 35.3% in 2060. Another important variable that indicates that the population is becoming older is the old-age dependency ratio, which is the number of old people (65 and above) relative to those aged 15 to 64. EU old-age dependency ratio is expected to rise from 31.6% in 2020 to 51.4% in 2060. This means that there will be only 2 working-age people, instead of 3.2, for each old person in the EU. That being said, health related challenges will have a huge impact in the world. The demand for primary health-care and long-term care will increase, as well as the required workforce and its training. Furthermore, the creation of elderly friendly environments will be necessary both to improve ageing quality and to allow active ageing, bringing social and economical benefits to the societies that adopt them.

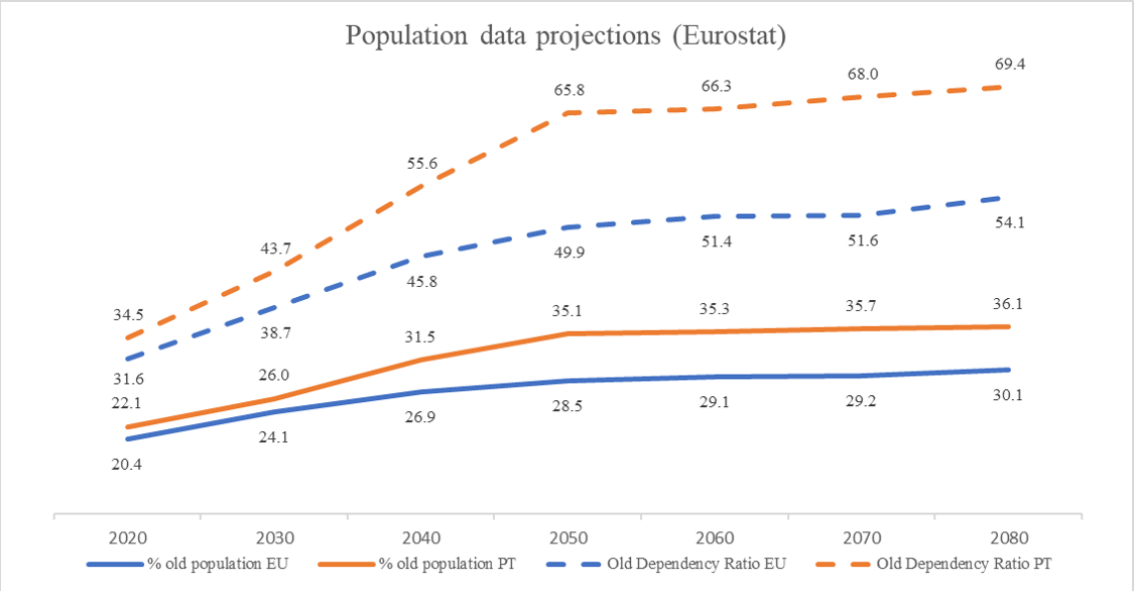


Figure 1.1: Demographic projections for the population according to Eurostat [1]. The percentage of old people (age $\geq$ 65) and the Old Dependency Ratio for the EU countries and Portugal are repesented in this Figure

According to the *World Report on Ageing and Health* by the World Health Organization (WHO), Healthy Ageing is the process of developing and maintaining the functional ability that enables well-being in older age [2]. Functional ability corresponds to the health related capacities that allow people to live the life their are willing to. The individual's intrinsic attributes, the environmental characteristics and the interactions between these two factors make up the functional ability.

According to the same report, nutrition is defined by the WHO as one of the key behaviours that influence Healthy Ageing. There are several physiological alterations that accompany ageing and can impact negatively the nutritional status. Sensory deterioration like decreased sense of taste and smell may cause reduced appetite [3], [4]. Moreover, vision, hearing and mobility loss, for example associated with osteoarthritis, may impact elderly people's capacity to prepare meals and shop for food. Furthermore, oral and gastrointestinal health conditions can also dictate the diet of an individual, sometimes leading to a state of malnutrition [5]. Difficulty in the chewing movement, gum inflammation and monotonous diets, low on quality and necessary nutrients, are conditions related to dental problems, while impairments related to gastric acid secretion reduce the absorption of iron and vitamin B12. Besides these physiological changes, ageing is also associated with psycho-social and environmental alterations related to isolation, loneliness [6], depression [7] and inadequate finances [8], which can have a major influence in dietary concerns. The combination of these trends increase the risk of malnutrition in this population. Although energy needs decrease with age, the need for most nutrients continues almost the same. There are some physiological factors that indicate that an individual is malnourished: reduced muscle and bone mass [9], which increases frailty; decreased cognitive function [10] and reduced self-care ability, which raises the risk of becoming care-dependent [11].

Malnutrition is a state that results from a dysfunction in the uptake or intake of nutrition.

It can be classified in undernutrition, overnutrition, specific nutrient deficiencies and imbalance caused by disproportionate intake. This topic is going to be further explored in Section 2.

According to the Global Nutrition Report, the number of overweight or obese adults is estimated to be 2.69 billion worldwide [12], which corresponds to approximately 34.9% of the world population. The WHO has stated that the total number of underweight adults amounts to 462 million worldwide [13]. In the elderly community, this prevalence varies greatly across the different population subgroups and social status. In the European countries, malnutrition prevalence is lower than 10% of the non-institutionalised elderly population [14], which are older people that live independently. However, in the same community, malnutrition has been reported to reach more than 50.0% of the population when considering nursing home residents and patients, both in acute care hospitals and in geriatric rehabilitation [15]–[17].

Besides the clear problems that malnutrition brings to individuals, there is also an economic burden to the Health Systems. Malnutrition and disease-related malnutrition (undernutrition) is associated with higher costs in healthcare services for community-dwelling or institutionalized patients in the elderly group. According to a review conducted by Abizanda et al. [18], which analysed different studies on the economical impact of malnutrition and related diseases, the cost of dealing with malnourished patients is higher than the cost associated with non-malnourished or even patients at risk of malnutrition (2000€ more per patient and year). This difference is explained by the higher use of healthcare resources such as medical consultations, hospital admissions and extended length of hospital stays. Another study reports that the costs associated with malnourished patients was more than twice that of well-nourished patients in the United Kingdom [19]. In the same country, the prevalence-based costs of dealing with malnourished patients in a hospital context amounts to approximately £3.8 billion which corresponds to 2.4% of the National Health System (NHS) budget for 2019. In the case of the Continental Europe countries, a total amount of €120 billion is spent on malnutrition (undernutrition only) patients [20]. Also, when comparing old and young malnourished patients, the costs of treating the former group are four times higher than that of the latter group [18]. The total worldwide estimated amount spent on all forms of malnutrition could reach up to $3.5 trillion per year, with overweight and obesity costing $500 billion [12]. Therefore, it is possible to conclude that malnutrition and its related diseases comprise a real problem, not only for the elderly community, but also for society, since it involves higher mortality and morbidity rates and high healthcare expenses.

In fact, malnutrition in older ages often goes undiagnosed until a late stage. So, in order to tackle malnutrition, there are some support and prevention approaches. In fact, there is a close relation between the diet and the risk of developing disorders like diabetes, cardio-vascular diseases, cancer and cognitive diseases [21]. For this reason, it became necessary to explore correct and clinically relevant nutritional recommendations – *Human Nutrition*. Moreover, it is important for the nutritional practice and research communities to designate the core nutritional processes involved. Under this scope, *Preventive Nutrition* and *Clinical Nutrition* become the pillars of the nutritional practice [21]. The former concept studies the way nutrients and food intake affect the risk of developing malnutrition-related diseases, both for individuals and populations. It often results in Public Health nutritional policies and campaigns, so that the levels of nutrition-related diseases are reduced for the population. On the other hand, *Clinical Nutrition* addresses the fields

of diagnosis and prevention, as well as the handling of changes in the nutritional and metabolic scene, related to both acute and chronic diseases and conditions caused either by excess or lack of energy and nutritional uptake. The nutritional actions taken on measurement, prevention or treatment of individuals are included in the scope of *Clinical Nutrition*. The latter is based on the interplay between food intake and catabolic processes that may be related to disease and ageing. This requires knowledge and science about metabolic disturbances and biochemical body composition and function under disease contexts. Dietary advice about food preparations and choices is pertinent for patients, relatives and care-givers, when preventing or treating malnutrition. As a matter of fact, according to the NHS, dietary assistance is the recommended way to avoid malnutrition. A balanced diet that comprises healthy foods is crucial for obtaining the necessary caloric and nutrients intake [22]. The dietary counselling is present under different contexts, namely Hospital Diet and Therapeutic Diet. Both demand knowledge about the individual nutritional requirements in order to develop an adequate plan. Other factors that are also considered in the dietary composition are local food habits and patterns [21], as well as food preferences, its price and availability [23].

Dietary planning can be addressed by dietitians and nutritionists. However, taking into account the previous information and the fact that technology is largely widespread in the healthcare field, there is the opportunity for meal recommendation systems to appear as a way to provide nutritional assistance. These information systems can be used to assist people in their decisions by providing a nutrition companion guide and will be analysed in depth in Chapter 2. Recommender systems for health applications must consider specific challenges, that can be divided in two groups [24]: algorithm and user challenges. The first challenge can be fulfilled by crossing information with clinical guidelines and health counseling from institutions such as the WHO [25] and the NHS [22]. In the case of more specific nutritional needs, suitable guidelines can also be consulted, such as those specific to the elderly population [26]. Regarding the latter, an information gathering phase is of crucial importance to deliver tailored recommendations. The system must collect enough information about the user's nutritional needs and food preferences (intial information collection) as well as the composition of his or her previous meals (continuous information collection), in order to give proper suggestions. Moreover, the continuous data must be acquired through cooperation with the user, which has been proven to be a difficult task [24]. The user needs to continuously report the meals taken, as well as give feedback about them. The interaction between the system and the user is crucial to keep user engagement.

1.1 Motivation

Human-computer interface is a key subject in the interaction and effectiveness of Information and Communications Technology (ICT) systems. User experience is highly dependant on the interface used. A clear example of such interface are the graphical user interfaces presented by mobile applications. These programs have seen an exponential growth in usage rate, mainly due to the fact that smartphones are largely widespread across developed countries. The number of smartphone owners in the United States has increased from 35% in 2011 to 81% of the population in 2019 [27], while it is estimated that the number of worldwide smartphone owners amounts

to 5 billion people [28]. Therefore, it makes sense to tackle the problem referred in the previous section using a mobile approach. Currently, there have been many efforts on creating mobile health applications for customer use [29]. A study from Research 2 Guidance [30] has estimated that there were almost 325 thousand mobile Health applications available for download on major app stores in 2017. According to a survey in the US adult population, 69% of the people reported to keep track of a minimum of one health indicator such as weight, exercise routine, diet or even symptoms [31] through a mobile app. There is also one report by the Pew Research Center stating that, although older generations have lower technology usage rates when compared to younger generations, the adoption and inclination to bring technology into the daily routine has increased more than any other generation [32]. Moreover, artificial intelligence (AI) in the healthcare field has also suffered a large growth. It is estimated that the total market size for AI in the healthcare sector could reach $6.6 billion in 2021 [33], which includes personal intelligent assistants.

Following this line of thought, *Fraunhofer Portugal Research Association*, a non-profit association that focuses its research efforts on active and healthy ageing solutions and works closely with the elderly population, has developed different systems to address this problematic, showing improvements in older people diet planning and nutrition, as well as the effectiveness of such systems [34]. Fraunhofer's main projects in this area are *CordonGris*[1], *LIFANA*[2] and *SousChef* [23], [34]. All of them are nutrition recommender systems that were designed with the elderly as their target audience. Coming back to the interaction between the applications and the user, this is made through a graphical user interface in most of the cases. However, the interaction between the elderly and their smartphones is often limited and not dynamic enough to motivate the user to continue to use it. This is where chatbots come in as a disruptive and more natural way of promotion interaction between the user and the system. Chatbots are created to simulate a natural conversation with a high degree of personification, being perceived as an advanced form of social cues in human-machine interaction [35]. They are conversational agents that use artificial intelligence algorithms to process natural language, and this is their main advantage: the interaction is made using speech or text. The study by Brandtzaeg and Følstad [36] analyses the underlying reasons for the utilisation of chatbots. The authors employ the *uses and gratifications* theory (U&G) to explain user motivation [37]. As stated, this theory tries to meet the cause and the way people use a given method to attain a specific goal. It claims that the use of a specific method depends on the gratification it will provide, both in terms of expectation and experience. The main assumption on which this theory is based is that users are goal-driven when selecting the manner to accomplish the goal. This assumption is widely applied in digital and electronic media. A similar study concluded that gratifications like the need for information, entertainment, self-expression and social interaction may be the motivators behind the choice of a digital media [38]. Brandtzaeg and Følstad concluded that productivity was the main cause to use chatbots, namely due to ease, speed and convenience associated with the use of chatbots. Also, they highlighted the fact that chatbots provide assistance and access to information, with a more natural way to input data to the system, facilitating the process of giving feedback [39]. The GUIDE project [40] concluded that speech-based interaction was the most favoured interface for the elderly population that is unfamiliar with technology.

[1] http://cordongris.eu/project/
[2] https://www.aicos.fraunhofer.pt/en/our$_w$ork/projects/lifana.html

There is also a large social component in the adoption of chatbots. Chatbots have already been used in contexts where preconceived ideas might hinder the processes, like the reporting of sexual harassment. Ideally these virtual agents are trained in such way that they have no preconceived ideas, thus making no judgement, which alleviates the shameful sentiment of the victim [41]. It is important that the system engages emotionally with the user since, according to a study by Xu et al., 40% of the enquiries are emotional rather than looking for specific information [42]. As a matter of fact, affective computing becomes very important when dealing with elder generations that have lived almost their whole adult life outside of the technological context. The three main reasons to consider this field in this case are the role that emotions play in cognitive and physical health, the lack of understanding of the metaphors used by technology and its intent, and the fact that people looking for social interaction tend to seek social contact even with virtual agents [43]. In addition, sentiment analysis can be performed to assess the emotional state of the user. This can become very useful to measure user motivation and as a feature to predict engagement with the system and improve user experience by guiding the changes that the system may suffer [44]. In this specific case, sentiment analysis can be used to identify food preferences and to obtain implicit feedback on meals.

The projects conducted in *Fraunhofer* do not implement any sort of conversational agents, whose application has proven to be beneficial for user experience [35], [39]. Besides the opportunities referred before, the utilisation of chatbots also brings economical advantages. These agents represent an easily scalable solution that greatly reduces the need for human resources [41] and associated costs. Thereby, a need has been identified, that could be tackled with the development of a domain-specific conversational agent capable of identifying user requirements, as well as receiving feedback information to lead the suggestions given by the recommendation system.

1.2 Goals

The main goal of this dissertation is the development of a conversational agent that can be integrated with a nutrition recommender system by using conversation as the means of interaction. At the end of the work, the foundations and a usable version of the chatbot should be attained.

The chatbot must be able to collect necessary user information and process it correctly, so that the data can be used by the recommendation engine that will consequently create the meal suggestions. This requires the implementation of Natural Language Processing techniques in order to parse user input into structured data, comprising of the user intention and the important terms and concepts in it, such as mealtimes or food items. For this task, a set of methods will be tried and explored in Chapter 5. Also, in order to complete this task, a proper dataset was collected and annotated.

In addition, a preference model for food items is also part of the goals of this work. With a view to enhance user interaction and immersion, the conversational agent is required to understand and transform a qualitative user input expressing preference into a quantitative metric that can be used by the recommendation system. Bearing this in mind, the system should be able to extract this information from user input using machine learning models. This tasks is addressed in Chapter 5 and employs a Sentiment Analysis tool to identify both sentiment polarity

and intensity.

The preference model works by providing a rating for each ingredient in the concerned recommendation system. However, due to the complex intrinsic semantic relations between food items, this poses as a challenge to be address. Therefore the development of an improved retrieval algorithm is also a requirement of this work. In light of this, food ontologies are a good resource of semantic information that can be incorporated into the chatbot. The encoding of this information into the chatbot and the revamped retrieval algorithm will be addressed in Chapter 5 and will use embeddings as knowledge source.

Being a chatbot, the system must have an interaction based on sequential conversation turns that should follow a correct logic. That being said, this work also requires a procedure that enables response prediction in order to maintain conversation. This task was also addressed in Chapter 5 and involved the experimentation of several approaches from simpler rule-based models to more complex deep learning models. This also required data collection.

Besides the mentioned tasks, the goals of this work also contemplate the evaluation of a final version of the chatbot. The evaluation should be made according to two different perspectives. A technical evaluation comprising performance metrics and a user judgement evaluation based on user opinion and testing. The former should be achieved during development while the latter should be attained by designing a usability test aimed at the evaluation of the principal mechanics and use cases of the chatbot. User-based evaluation and validation is addressed in Chapter 6 and includes the analysis of questionnaire answers for chatbot evaluation after volunteer testing.

In light of this, the proposed agent will facilitate the interaction between the recommender system and the user, allowing for an easier and more intuitive way to explore the wide range of functionalities presented, which ultimately will increase the usage rate of the system. Given this, it is expected that the nutritional status of the user will be improved, resulting in a more active and healthier life.

That being said, the final system will be comprised of a conversational agent that employs Natural Language Understanding and Dialogue Management algorithms capable of creating structured information requests to query the already implemented recommender system and return appropriate responses. The system will serve as the interaction interface between the user and the recommender system.

1.3 Contributions

This work provides the following contributions:

- Comprehensive analysis of Lifana nutritional recommender system, including meal plan structure and available food data that ranges from recipes to ingredients
- A detailed exploration of the Rasa framework for chatbot construction, including Natural Language Understanding and Dialogue Management pipelines

- A set of pre-trained word embeddings based on ConceptNet Numberbatch that were fine-tuned to incorporate semantic knowledge from a food ontology, obtained using Retrofitting. These embeddings encode the principal relations between ingredients and can be used for several tasks
- An algorithm for database entries retrieval given a query word, based on word embeddings similarity. This algorithm is able to distinguish between individual ingredients and groups that relate to a given ingredient and was validated through user testing
- An algorithm that performs preference modelling from text input based on sentiment analysis. It extracts the sentiment of a sentence regarding an ingredient and transforms it into a numeric rating based on polarity and intensity
- An annotated dataset for Natural Language Understanding tasks, namely intent classification and entity recognition under the scope of this work
- A dataset of conversational turns that enables the training of a Dialogue Management model
- A comparative analysis between usability and user experience questionnaires as a way to evaluate chatbot interaction
- A conversational agent capable of performing the interaction between a recommender system and a user and the framework for further improvements

1.4 Document Structure

The present document is organised in 6 additional chapters. Chapter 2 covers the theoretical background concepts that may be necessary to fully understand the document. In particular, it explores the concepts of malnutrition and related diseases, as well as the diet role in the presented disorders. It also presents the recommendation system developed by *Fraunhofer*, along with its features and architecture, which will be the base of the proposed work, and the theoretical concepts of Natural Language Processing that allow the construction of chatbots.

Chapter 3 provides an analysis of the functioning and possible ways to develop conversational agents, including available tools and evaluation methods. Moreover, the state-of-the-art and literature review on these topics is also presented.

The fourth chapter introduces the methodology and planning of the dissertation work. Also, it describes the initial available resources and the used framework, as well as the Use Cases that will be contemplated in the chatbot development.

Chapter 5 details the experimental setup and evaluation used to overcome the challenges. Namely, it mentions the collection of both the Natural Language Understanding and Dialogue Management datasets, the pipelines tried and selected and both the development of the preference model and the food retrieval algorithm.

Chapter 6 describes the human judgement evaluation made to the final chatbot by means of user testing. It explores the designed protocol and the results obtained for user experience and usability.

To sum up, Chapter 7 presents the conclusions taken from the development. It analyses the methods used to reach the goals described in Section 1.2 and presents some future lines of work that may be employed for further improvement or the inclusion of additional features to the conversational agent, providing some suggestions.

Chapter 2

Background

In order to properly construct a conversational agent that can meet the requirements referred in Section 1.2, it is important to understand the underlying concepts of the domain, as well as the scientific and technical knowledge about chatbots and language processing. In this chapter, a more in-depth description of the referred ideas will be presented to allow for a better comprehension of the following work.

2.1 Malnutrition

Malnutrition is a state that results from lack of uptake or intake of nutrition, and usually leads to altered body composition and body cell mass which are normally the underlying cause of diminished physical and mental function and impaired clinical outcome from disease [45]. According to the World Health Organization, there are several types of malnutrition [13]:

- **Undernutrition**: There are three types of undernutrition which is defined as the lack of nutrients.
  - *Wasting*: The state characterised by low weight-for-height is known as wasting. Recent and severe weight loss are usually related to wasting, mainly due to not having enough food to ingest and/or having an infectious disease, like diarrhoea, which leads to weight loss.
  - *Stunting*: This case corresponds to low height-for-age and is generally associated with chronic or recurrent undernutrition [46]. It is most common in children and is normally associated with low socioeconomic backgrounds, poor maternal health and nutrition, chronic and recurrent illness, incorrect dietary components and poor quality early life care. It is estimated that approximately 162 million children under the age of 5 years are affected by stunting [46].
  - *Underweight*: An individual (often children) is considered to be underweight when they have weight-for-age compared with reference values. The individual may also be wasted, stunted or both.

- **Micronutrient-related malnutrition:** Deficiencies in micronutrients (minerals and vitamins) intake lead to a state of malnutrition. These micronutrients are essential for the internal production of enzymes, hormones, and other vital substances for correct growth [13]. It becomes important to take into account these micronutrients in the diet.

- **Overweight and Obesity:** These states happen in the case of an individual being too heavy for his or her height. It is a serious condition that has several consequences for health. The main way to diagnose and classify overweight and obesity is the Body Mass Index (BMI) which relates the weight and the height of an individual [13]. More specifically, it corresponds to the ratio between the weight (kg) of a person and the square of his/her height (m^2) as seen in Equation 2.1. The reference values for the BMI, according to the WHO, are shown in Table 2.1

$$BMI(kg/m^2) = \frac{weight(kg)}{height^2(m^2)} \tag{2.1}$$

Table 2.1: Body Mass Index reference values according to the World Health Organisation [47]

BMI	Nutritional status
Below 18.5	Underweight
18.5–24.9	Normal weight
25.0–29.9	Pre-obesity
30.0–34.9	Obesity class I
35.0–39.9	Obesity class II
Above 40	Obesity class III

The balance between the caloric intake and the energy wasted during physical activities is what regulates the retention and accumulation of fat. The consumption of foods and drinks that are high on fat and sugar and the lack of physical exercise are asserted as key factors leading to the global increase of obesity and overweight in people.

- **Diet-related noncommunicable diseases:** Another group of conditions that result from unbalanced or insufficient intake and uptake of nutrients and micronutrients are the diet-related noncommunicable diseases (NCD) [13]. Diseases such as specific cancers, diabetes,

and cardiovascular diseases have a poor diet and unhealthy nutrition habits as aggravating and underlying factors [48]. As a matter of fact, NCDs are the cause of death of 41 million people annually, which corresponds to 71% of all deaths globally [49]. More precisely, 17.9 million deaths are due to cardiovascular diseases, which are followed by cancer that causes 9.0 million deaths/year and diabetes (1.6 million) [50].

2.1.1 Malnutrition and Diet

It has been proven that diet takes a major role in the likelihood of developing malnutrition [51]. The world food industry has been focusing the efforts in reducing costs and increasing efficiency so that global hunger is reduced. Bearing this in mind, the production of high-energy foods and drinks grew and with this, the local diets have been displaced and altered, decreasing in diversity and quality [52], [53]. As a matter of fact, the access to healthy foods, rich in micronutrients and minerals, have not improved equally for everyone, while unhealthy foods have become cheaper and more accessible and widespread [54]. Also, the global demand for meat, dairy products and processed foods has increased massively [54]. The dietary choices are influenced by several factors such as culture, skills, economics and politics, as well as time availability and information [52].

According to the data available in the Institute for Health Metrics and Evaluation of the University of Washington [55], diets that are low on whole grains, fruits, vegetables, nuts and seeds, and high on sodium are the ones that contribute the most to the burden of diet-related diseases.

2.1.2 Healthy diet

The idea of a universal diet that fits all the population is unrealistic, since it greatly depends on the characteristics of each individual, such as gender, age, lifestyle, and the level of physical activity, as well as political and cultural context and food availability. Nevertheless, the fundamentals of a healthy diet remain the same and must be considered when addressing a meal plan [56]. According to the information of the WHO [56], an adult healthy diet includes:

- Fruits, vegetables, legumes, nuts and whole grains;
- A minimum of 400g of fruit and vegetables per day [57];
- A maximum of 10% of energy intake from free sugar[1] consumption. However, the ideal amount is 5% [58]. The maximum daily caloric intake should be around 2000 calories.
- A maximum of 30% of energy intake form fats. Not only that, but the consumption of unsaturated fats is preferable when compared to saturated fats and trans-fats. Moreover, the recommended value of energy obtainment trough saturated fats is 10% and through trans-fats is less than 1% [59].
- A maximum of 5g of salt which should be iodised [60].

[1]Free sugars are all sugars added to foods or drinks by the manufacturer, cook or consumer, as well as sugars naturally present in honey, syrups, fruit juices and fruit juice concentrates.

Another important indicator for a healthy diet is the recommended daily dose of micro and macronutrients [61] observed in Tables 2.2 and 2.3. A planned diet should meet these requirements.

Table 2.2: Minerals and vitamins and their daily nutrient reference values [61]

Vitamin A (μg)	800	Chloride (mg)	800
Vitamin D (μg)	5	Calcium (mg)	800
Vitamin E (mg)	12	Phosphorus (mg)	700
Vitamin K (μg)	75	Magnesium (mg)	375
Vitamin C (mg)	80	Iron (mg)	14
Thiamin (mg)	1.1	Zinc (mg)	10
Riboflavin (mg)	1.4	Copper (mg)	1
Niacin (mg)	16	Manganese (mg)	2
Vitamin B6 (mg)	1.4	Fluoride (mg)	3.5
Folic Acid (μg)	200	Selenium (μg)	55
Vitamin B12 (μg)	2.5	Chromium (μg)	40
Biotin (μg)	50	Molybdenum (μg)	50
Pantothenic (mg)	6	Iodine (μg)	150
Potassium (mg)	2000		

Table 2.3: Daily reference intake values for macronutrients [61]

Nutrient or energy	**Daily Reference Value**
Energy	8400kJ/ 2000 kcal
Total fat	70 g
Saturates	20 g
Carbohydrate	260 g
Sugars	90 g
Protein	50 g
Salt	6 g

A healthy diet follows the aforementioned recommendations and should therefore be preserved. In order to do so, the WHO also suggests some good practices to follow, divided by categories.

- Fruit and Vegetables:

  The intake of the recommended amount of fruit and vegetables is very important since it constitutes the only source of fibre in the diet, which is fundamental for adequate gastrointestinal function, as well as a good source of micronutrients such as vitamins and minerals. The WHO recommends the inclusion of vegetables in every meal, the consumption of fruits as snacks, in season fruit and vegetables consumption (supporting sustainable nutrition) and varying the intaken fruits and vegetables [56].

- Fats:

  Increased fat consumption is directly related to unhealthy weight gain and also cardiovascular diseases. In this case, the WHO recommendations are the reduction of saturated and trans-fats and its substitution by polyunsaturated fats such as soybean, rapeseed, sunflower or corn. This can be achieved by giving preference to cooking methods such as

steaming or boiling instead of frying; replacing butter in the diet by the polyunsaturated fats referred above; reduce the dairy fat consumption and increasing lean meat uptake and limiting the ingestion of baked and fried foods and pre-packed snacks, since they contain a large amount of trans-fats [56].

- Salt, Sodium and Potassium:

  There is an excessive intake of salt. Globally, this translates to high amounts of sodium in the diets. However, potassium, an essential element in cellular function, is often underconsumed. This unbalance usually leads to high blood pressure and increased risk of heart disease and stroke. If the recommended amount of salt was taken into consideration (5g), a total of 1.7 million deaths could be prevented annually. Salt intake may be decreased by limiting the addition of salt while preparing meals, limiting the amount of sauces and snack that are high on salt content and also a better selection of food that is low on sodium. Furthermore, the consumption of potassium can help decrease the harmful effects of sodium and can be obtained in fruit and vegetables [56].

- Sugar:

  The excess of sugar consumption presents negative consequences for health. Namely, tooth decay (dental caries), unhealthy weight gain, high blood pressure, an imbalance in serum lipids and diabetes. The recommendations of the WHO regarding sugar reduction are the limitation of added sugar foods and drinks and the increase of the consumption of fruits and vegetables to replace sugary snacks [56].

2.2 Lifana Nutrition Recommender System

As stated in Section 1.1, *Fraunhofer Portugal* has developed a nutrition recommender system to help weekly meal planning mainly amongst the elderly population [23], [34]. The system is able to define a weekly meal plan according to the user preferences and using some heuristics and restrictions that will be further explained. Currently, the system is composed by a central cloud server and a mobile application that serves as the interface for user interaction. In this section, the details of the recommender system will be provided in order to understand the requisites of the interaction system.

2.2.1 Cloud server component

The key component of the recommender system is the cloud server that is responsible for both the data handling process and the generation of recommendations [34]. The first task comprehends the process of data storing and retrieval, by making it accessible using web service application programming interfaces (API). Information from external sources can also be integrated in the information storage unit. An example that is given in the system report is the collection and storage of physical activity information, collected through a *Fitbit*[2] device, from the *Fitbit* cloud services. The referred data is important to create fitter recommendations, namely the personal

[2]http://www.fitbit.com/eu/home

demographic information, the activity data and nutritional information from the considered databases (explored in Section 2.2.2). Regarding the task of recommendation generation, the higher computational capacity of the cloud server is an advantage against the processing power of mobile devices, which makes the server a better option. This way, the recommendations can be induced by APIs on the mobile devices.

2.2.1.1 Recommender engine

The recommendation engine implemented in *Lifana* is a content-based recommender system and makes use of information retrieval techniques. As a matter of fact, recommender systems can be implemented in three possible ways.

- **Content-based:** These models match the the information provided by the individual user with the content information in order to produce the recommendations [62]. These systems ignore the recommendations provided to other users as well as their preferences. More often than not, these models track the patterns of the users [63] to predict the next steps and also ask for user feedback input so that it can create a baseline recommendation model. It would recommend items that are similar to the items that the user prefers. These models suffer from several problems, namely the limited content analysed by a new user (cold-start problem), overspecialisation that leads to the recurrent recommendation of the same content and the sparsity of the data [64];
- **Collaborative filtering:** As the name implies, these models are based on preference similarities between users [62]. The feedback from other users is taken into account to generate recommendations to the end-user. This way, the system creates user profiles based on their tastes and uses them to match other users, sharing the recommendations between them [65]. Collaborative filtering methods are limited by the cold start, in which there are no sufficient user ratings to match similarities between them, data sparsity and scalability problems [62].
- **Hybrid approaches:** Models that apply both techniques in order to achieve higher quality recommendations and counter the liabilities of the other two models [62].

Lifana works by performing three main sequential steps [23], represented in Figure 2.1.

1. **User's nutritional requirements estimation** using the guidelines provided by the WHO, the appropriate number of meals per day and the personal user information referred in the introduction of Section 2.2.1.
2. **Selection of the food items** that are going to be part of each meal according to the user preferences and needs and also taking into account the history of previously suggested meals (to increase dietary variance). The available database of possible recipes is consulted to define the candidate recipes for that week. Then, a two step filtering process is conducted in order to select the final plan. (1) The candidates are filtered according to the restrictions imposed such as limiting the repeated recipes in the same week, removing options that include comfort food and ingredients that cause allergic reactions on the user or

combinations that are not nutritionally relevant. (2) The second phase selects the most suitable options between the remaining candidates. In this phase a score is calculated using the different heuristic criteria that are defined. The plan receives a score for each criteria and the final score is then considered to make the decision. The heuristics were defined by a nutritionist and consist in criteria that favours the user experience or facilitate the process of meal preparation. For instance, plans that have the same soup in four consecutive meals are favoured since the preparation is easier. Moreover, plans that have meat dishes for lunch and fish dishes for dinner are preferred. This phase also considers the user preferences regarding ingredients or meals.

3. **Meal scaling** to match the required energetic requirements. This stage uses the caloric and nutritional information of the ingredients to scale the recipe. The scaling occurs only to a designated group of ingredients. The system performs a weighted scaling, giving preference to the ingredients that have a higher energetic value, so that the size of the meal does not change excessively.

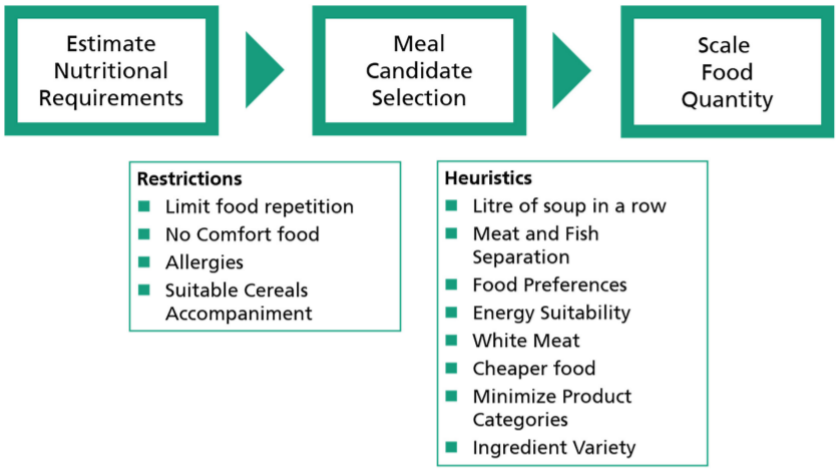


Figure 2.1: Lifana recommender system sequential stages [23]

All these steps were designed with the assistance of a nutritionist.

2.2.1.2 Mobile application

The interface between the recommender system and the user is a mobile application (app) that is divided in three parts: meal plan, grocery list and activity monitor. The app also has a profile that groups information related to the user. The interface of the profile can be seen in Figure 2.2.

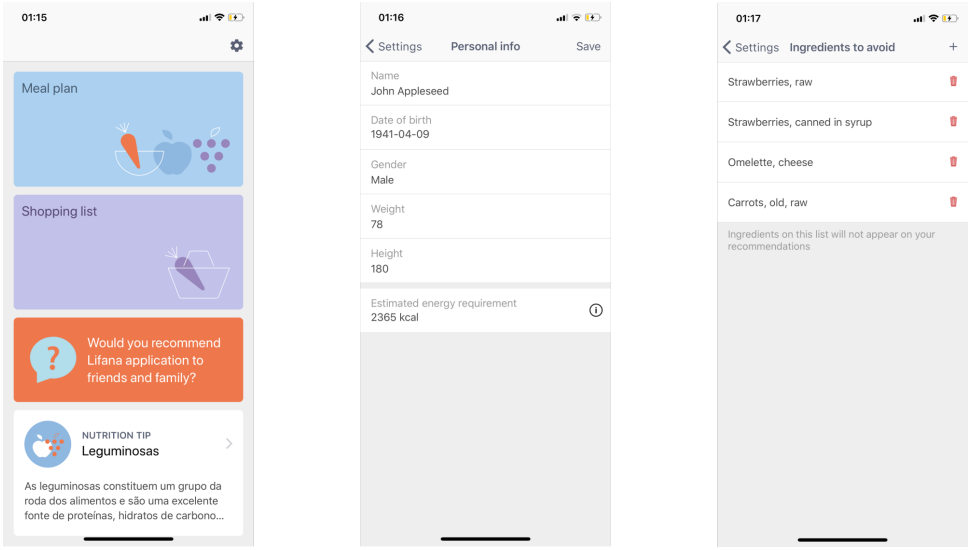


Figure 2.2: Left: Lifana main screen; Centre: User profile; Right: Food Restrictions

- **Meal Plan** The core part of the whole system that has two tasks: generate and expose the recommended meal plan and also track everything the user eats. In order to fully cover the eating habits and the nutritional status of the user, a manual eating diary where the user can input the content of every meal he or she consumes was implemented. In addition to this, detailed information about the recipes or ingredients can be consulted, more specifically their nutritional information and energetic content. Moreover, feedback about the ingredients and the meals can be given in this part of the app.

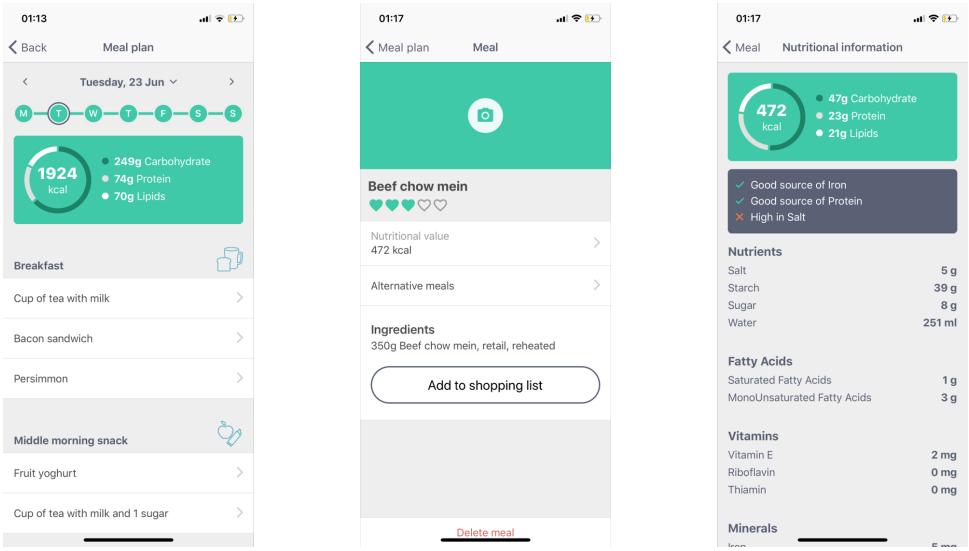


Figure 2.3: App screens for meal plan, recipe information and nutritional content

- **Grocery list**

  This part of the app has the goal of guiding and assisting the shopping process of the user. Planning the shopping list can be a very useful task when deciding and determining the meal plan for a whole week. This way, the user can assure that all the necessary ingredients

and components are available. The grocery list is connected to the plan for the week and may automatically add all the necessary ingredients of a recipe to the list. Furthermore, the addition of individual ingredients is possible by a search in the database.

- **Activity Monitor**

  The integration with activity-measuring devices allows the system to calculate and show to the user some metrics that are relevant for a healthier lifestyle. Also, the physical activity is one of the factors that is considered when calculating the user needs. Therefore, the app connects to an activity-monitoring device such as the referred Fitbit and collects data from it. This data can also be inserted manually in the application.

All the interactions between the app and the user, such as searching for recipes or ingredients, providing feedback, inputting the physical activity data or even asking questions about the recipes can be changed to a more immersive and natural approach by using a conversational agent, which is the focus of the work reported in this dissertation.

2.2.2 Knowledge bases description

The already implemented recommender system explained in detail in Section 2.2.1.1 relies on a database that was created on the national food composition database (FCDB) provided by PortFIR[3] and licensed to EuroFIR[4]. This database relies on information from a food ontology, thus allowing for complete knowledge extraction from food and its composition. The referred ontology is LanguaL[5], which is a multilingual thesaurus system that employs a facetted classification method. Each food is described by a set of standard, controlled terms chosen from facets characteristic of the nutritional and/or hygienic quality of a food, such as the biological origin, the methods of cooking and conservation, and technological treatments. It uses a numeric coding in which each code represents a similar term in every language, thus making it language-independent.

The recipes that are used by the Lifana system were built on top of this information by mixing ingredients and quantities. In addition, the nutritional composition and the type of ingredient, as well as the meal time are also taken into account in the recipes. However, it is worth noticing that these recipes contain neither the cooking method nor the instructions for preparation. In Figure 2.4 it is possible to observe an example of a recipe stored in the database, while in Table 2.4 the numbers of generated recipes are shown.

[3] http://portfir.insa.pt/
[4] https://www.eurofir.org/
[5] https://www.langual.org/Default.asp

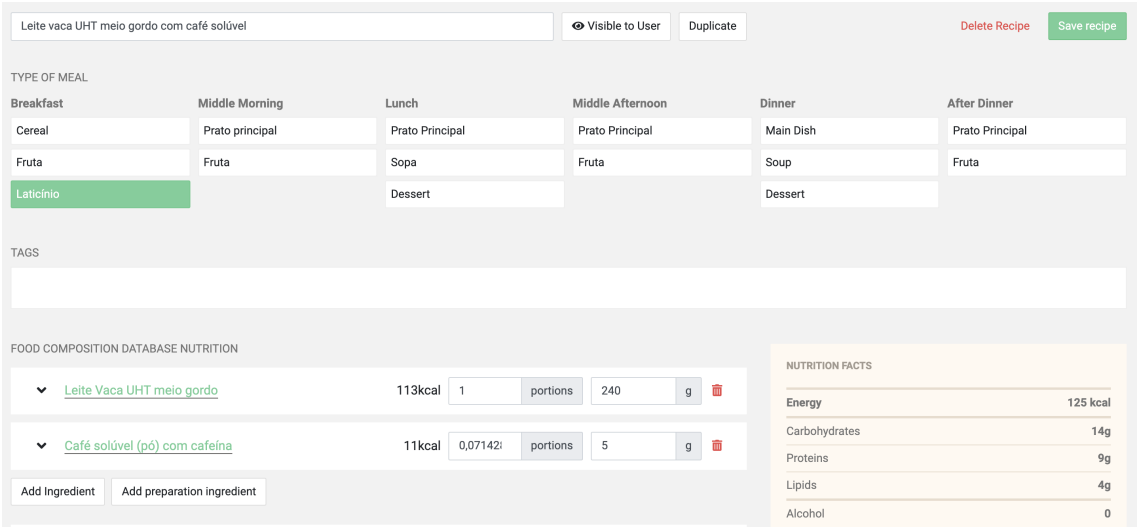


Figure 2.4: Example of the food database used by the Lifana system

Table 2.4: Number of created recipes for each meal time [23]

Meal of day	Meal division	Number of recipes
Breakfast	Cereal	145
	Fruit	31
	Milk or milk derivate	10
Middle morning	Main dish	149
	Fruit	31
Lunch	Main dish	194
	Soup	18
	Dessert	38
Middle afternoon	Main dish	149
	Fruit	31
Dinner	Main dish	190
	Soup	18
	Dessert	38
After dinner	Main dish	149
	Fruit	31

That being said, the full comprehension of the structure of both the database and the food ontology is a foremost task for the correct communication between the recommender system and the database, as well as for the construction of the conversational agent. The agent must create queries that can be understood by the system and that allow for the retrieval of the required information.

2.3 Natural Language Processing

Natural Language Processing (NLP) is the field that studies the computational algorithms that take human language as input and transforms it into structured data. It comprises various mechanisms, from fundamental ones like attributing parts of speech to words to more complex tasks as question-answering and machine translation. These tasks often require data mining in texts – text mining – which is used to extract the information from text samples and use it in other applications [66].

One of the main differences between computer language and human language is the ambiguity present in the latter. Ambiguity corresponds to the event of having different interpretations for the same input and this is patent across all human languages and its different levels [66]. This property allows for a huge degree of variability in the data, along the levels of semantics and syntactics [67]. Also, the phenomenon of synonymy enables the use of different words with approximately the same meaning, increasing the semantic variability [68]. Moreover, syntactic variability can be observed in phrases holding the same meaning while being constructed in different ways, such as passive and active voice or possessive case [69]. These are the main challenges that NLP has to surpass in order to produce correct linguistics models that are capable of interacting with humans through their language.

2.3.1 Approaches to Natural Language Processing

There are two main approaches to NLP: rule-based or knowledge-based and statistical or machine learning based [66]. These approaches are not exclusive as there are models that employ algorithms from both. There is no current evidence that one model works better than the other when considering specific domains, although overall the statistical methods are superior in terms of results and scalability.

2.3.1.1 Knowledge-based NLP

This first way to address NLP makes use of information sources about the domain. Two types of information may be used in this case: knowledge about the language used and how information is usually presented, and real-world knowledge about the domain. The first ones are rule-based models, since they employ rules like linguistic patterns. The other kind of system benefits from specific cases of the domain [70]. For example, in this work, it would be useful for the system to know that *egg* is an ingredient and *boil* is a cooking method.

These models work in practice due to the fact that the rules that they apply and the information that supports them are formally, empirically and logically real. Also, these models do not need to go through a training phase on large datasets. However, hand-crafting the rules that define them usually consume a fair amount of time and resources. Additionally, pure knowledge-based systems can become rigid and less robust when dealing with more complex tasks like question-answering [71].

2.3.1.2 Statistical or Machine Learning based NLP

On the other hand, machine learning based or statistical NLP models are founded on computing algorithms that take advantage of labelled or and known examples or mined features from unlabelled data to learn how to classify and proceed with unseen ones. For this purpose, the model needs to be trained on a fairly sized dataset of examples without the necessity of domain-specific knowledge [72]. The essential assumption that these models are based on is that the frequency of the phenomena of interest is enough to build a strong statistical representation [66].

In order to have a good statistical representation of the data, a large amount of labelled data is required, which can be hard to obtain [73]. Normally, data labelling is an expensive and difficult process, not only because language data is very sparse, but also because some domains require specific expertise [74]. Machine learning models can also suffer from overfitting, which happens when the model excessively adjusts itself to the training data, becoming unable to correctly process new information.

2.3.2 Natural Language Processing tasks

With a view to extract relevant information from text samples, a pipeline of data mining and processing tasks should be followed. This way, computers are able to use the human language and produce the required outputs. The following subsections explore the main steps used in NLP for data preparation and feature extraction.

2.3.2.1 Document and Sentence Segmentation

The first step when analysing text data is to split the document into different sections, according to the logical structure. This step is followed by section segmentation into elementary blocks that deal with the same topic [75]. These are the standard unit of analysis in NLP systems and very often correspond to sentences. This process occurs using a model that examines syntactic cues such as punctuation marks and upper-case letters [76]. Also, there are some systems that apply similarity measures to group some of these blocks [75].

2.3.2.2 Tokenisation

The elementary blocks are then segmented once again into tokens which are the smallest element of the input data. In this case, tokens can be words, numbers, punctuation marks or even groups of words that should not be split like city names (eg. San Francisco or New York) [76]. Thus, the tokenization process should take into account related word dependencies and specific domain knowledge as well as the handling of morphosyntactically relevant linguistic specificities. Naturally, tokenization requires either a trained model or a rule-based model, for which domain-specific knowledge and grammar rules are necessary [77]. Furthermore, tokenization algorithms are dependant on the language. Different idioms have distinct grammar rules, including orthographic and lexical particularities, and tokenization may not be as tacit as it in the english language. A clear example of that are the oriental languages which have no explicit word

boundaries [78]. It is also important to mention that it could be advantageous to remove stop words such as "*the*" and "*a*", which give little or no information. After obtaining the tokens it is possible to label them with a named entity, allowing for the recognition of the different semantic classes present in the sentence. This operation is called ***Named Entity Recognition*** (NER) and is one of the key procedures in information retrieval and text mining tasks [79] (see example in Figure 2.5). Under the scope of this work, NER would allow the identification of the distinct components of a recipe such as the ingredients or the mealtime.

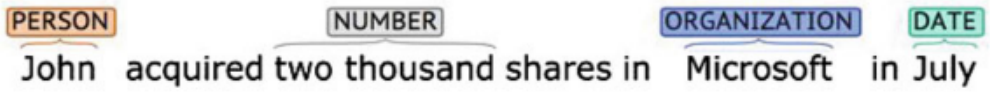


Figure 2.5: Named Entity Recognition example obtained using the StanfordNLP toolkit [80]

2.3.2.3 Part-of-Speech Tagging

The following step in the NLP pipeline is the Part-of-Speech tagging (PoS tagging) in which a PoS is assigned to each token. A PoS tag results from the lexical analysis of an element in a sentence. This process is considered to be a sequential labelling problem [78]. The results obtained from PoS tagging can become very important in the extraction of information for the specific context [66]. For instance, in a recipe for boiled eggs the system should recognise the word "*boil*" as a verb that is referring to the noun "*egg*". In this case, the verbs have a high probability of being related to cooking methods, while nouns are mostly related to ingredients. An example of a tagged sentence can be observed in Figure 2.6

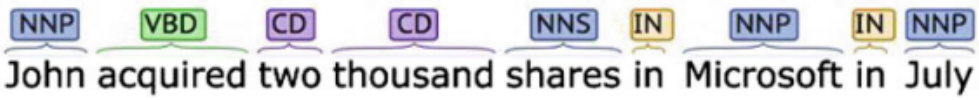


Figure 2.6: Penn Treebank Project [81] part-of-speech tagging [80]

2.3.2.4 Word Normalisation

After the process of tokenization, it could be necessary to normalise the data so that it becomes easily processed. Stemming can be used for the referred purpose. The goal of this mechanism is to group similar words according to their meaning (find the stem of each word). For instance, "*stir*", "*stirs*", "*stirred*" and "*stirring*" have similar semantic origin. The best known stemming approach is the Porter algorithm. This stemmer contains 60 suffixes, a contextual rule that guided the removal of the suffixes and 2 rules for word recoding [82]. After the Porter stemmer all the words from the given example would be transformed to "stir". More examples can be seen in Table 2.5

Table 2.5: Example of stemming using the Porter algorithm [83]

Original words	**Stems**
studied, studying, student, studies, study	studi, studi, student, studi, studi
miner, mining, mine	miner, mine, mine
vegetable, vegetarian, vegetate	veget, vegetarian, veget
eating, ate, eats, eater	eat, ate, eat, eater,

Word normalisation can also be performed by transforming the words into their "dictionary form" or lemma [84], instead of removing the suffix. The first approaches to lemmatisation used lookup tables for substitution [85], which worked very well for complex word forms such as the forms "worse" and "worst" of the word "bad". More recently, approaches using both lookup tables and rule-based systems based on the combination of hand-crafted features and machine learning [86]. This process often requires the part of speech tag in order to produce better results, since the transformation that is going to be applied to each word can change in accordance with morphological function [87]. Additionally, lemmatisers can be developed for a specific domain, increasing the quality of the results produced, like the work presented in [86].

2.3.2.5 Parsing

Syntactic information is extracted by the process of parsing. The result of this process is a structured tree that represents the grammatical relationship between the several components of the sentence [78]. According to the completeness of the task, it is possible to distinguish shallow parsing from full parsing. The first method tries to identify groups of words that usually go together. In order to do so, the algorithm focus on a "head" word of a given part-of-speech from which it can create the structure. On the other hand, full parsing tries to determine the full structure of a sentence by adding the relationships between the groups previously identified [66]. Moreover, full parsers can be divided in two classes: constituency parsers and dependency parsers. Whereas the first group focus on dominance and precedence and phrasal relationships in the sentence, dependency parsers discern and classify word-to-word relationships (see Figure 2.7. In the latter case, the labels produced are, for instance, *subject* or *object* and therefore dependency parsers are regarded as a more direct approach to the semantics of a sentence [66].

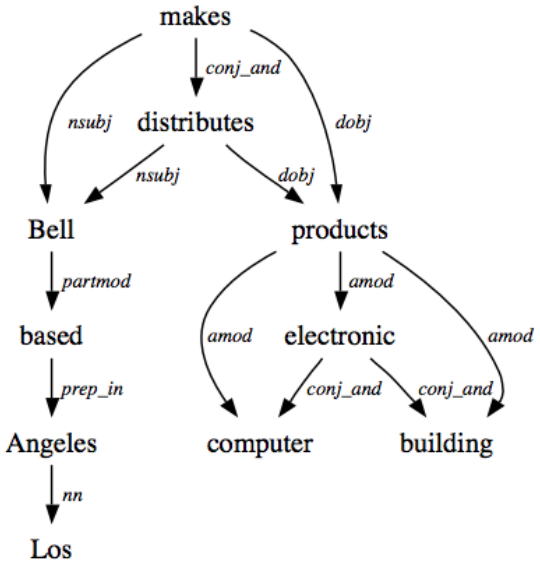


Figure 2.7: Dependency parsing example. The tree structure represents the grammatical dependencies between the token of the sentence: *Bell, based in Los Angeles, makes and distributes electronic, computer and building products* [88]

2.3.3 Natural Language Processing based on embeddings

In the previous section, the techniques used to extract textual features from the corpus have been described. However, more recently, approaches based on a vectorised representations of words have appeared. These methods try to mathematically map each word (or phrase) into a vector space, that also considers the relationships with other words [84]. This vector is an approximate way of representing the meaning of the word and it is supposed to maintain the relative relationship between the words. These representations could be generated from text data annotated with semantic and syntactic relationships. However, this kind of data is very hard to obtain (it requires a very specialised workforce, such as linguists), making it virtually impossible to generate a big enough dataset. In the light of this limitation, unsupervised methods appeared as the main ways to generate word vectors. There are two main ways to obtain embeddings which will be explored in the following sections.

2.3.3.1 Static Word Embeddings

These models are based on the premise that words that co-occur should have similar semantics. This way, it is able to capture the contextual relationships of words that appear close to each other in the sentences. Word vectors are created such that words that share contexts in the text are positioned close to one another in the vector space.

A well known algorithm used in this process is *Word2vec*. This model uses a neural network to process a document-based vector model to generate a word-based vector model [89]. This specific model consists in a neural network with two shallow layers and can be trained using

two different methods: Continuous Bag-of-Words (CBOW) or Continuous Skip-Gram. In the first approach (CBOW), the models tries to predict the current word using a window of words that make up the context. In this case, the order of the words in the window is not taken into account. On the other hand, the second approach (Skip-Gram) works by predicting the surrounding context window using the current word. It is considered that the Skip-Gram produces better results, though being slower.

It is possible to find pre-trained lexicons created through the application of Word2vec to huge corpora (such as the Google News Dataset). These lexicons have the advantages of not requiring training sessions and they are generic enough to capture the meanings of the words. However, this can be seen as one of their greatest disadvantages, which is the fact that they were not trained specifically for a given domain nor context and may not represent well enough the semantics of that domain/context. The words in a specific domain can be grouped differently from the texts where the model was trained. In order to fight this problem, it is possible to use this technique to create word representations for specific corpora, but it is important to have a large collection of data to achieve good results. Another drawback of these methods is the fact that the embeddings are static regardless of the context. This means that, independently of the context of a sentence, each word will have the same representation – the context is not taken into account after the training. For example, in the sentences: *"I love apple pie"* and *"I want to buy an Apple laptop"*, the vector for the word *apple* will have the same values even though they possess totally distinct meanings.

2.3.3.2 Pre-trained Language Models

More recently, in 2018, some investigators produced models that use a dissimilar approach in order to tackle the lack of context in the embeddings. These models are called language models and they employ a self-attention mechanism to change the embeddings of each word according to its context. In this case, the same word in different contexts will be represented by a different vector. Moreover, they allowed for a major performance improvement in several NLP tasks such as Question Answering or Natural Language Inference. The most popular language model is Google's BERT (Bidirectional Encoder Representations from Transformers) [90]. As stated above, these models apply a self-attention architecture, namely the Transformer module, also developed by Google [91], which is responsible for learning the word (or sub-word) contextual relations in a text. This architecture is composed by an encoder and a decoder. In BERT's case, only the encoder will be considered when using it, since the goal is to produce a language model. The encoder-decoder pair is used while training, which happens in a bi-directional way, meaning that each sentence is analysed under no preferential direction. This allows for a deeper understanding of the word context, both to the left and to the right of it. For a given input sentence, BERT works by firstly embedding the tokens and then processing them using a neural network. This pipeline outputs a sequence of vectors in which each one matches an input token.

Another key aspect about BERT is its training process. BERT is trained using two different approaches: Masked token prediction and next sentence prediction. This way, it is able to understand both the context of each word, as well as the context of the sentences in the document. The

creation of transformer models in NLP is regarded as the equivalent of the ImageNET for computer vision. It has drastically improved almost every task related to text processing. Coming back to the example given above (*"I love apple pie"* and *"I want to buy a Apple laptop"*), this model will no longer generate the same embedding for both instances of the word *"apple"*, since they are in different contexts. Furthermore, BERT is also capable of generating an embedding for a whole sentence embedded into one special __*CLS*__ token, that captures the meaning of it. This can be very useful in chatbot building as it will be discussed in the next chapter. These models are usually applied through transfer learning and fine tuning. The premise is that if a model is trained in a big enough corpora, it would be able to generalise for virtually every context.

2.4 Summary

In this chapter, the information required to understand the scope of the work and some applied techniques were explained. The next chapter explores the approaches currently used for chatbot construction and validation, including available tools and examples about health-related conversational agents.

Chapter 3

State of the Art

This chapter introduces the main approaches taken to build and evaluate chatbots. The first section will explore the concepts that are inherent to the construction of such an agent. The subsequent sections touch chatbot construction tools and evaluation methods. The chapter ends with the exploration of some health-related examples.

3.1 Chatbots and Communication

Chatbots are conversational agents that use Artificial Intelligence (AI) methods in order to communicate with people using human language [92], making them very useful tools in human-machine interaction. These applications make use of NLP and text mining algorithms and processes to achieve the referred goal. Also, chatter bots usually have text-to-speech and speech-to-text modules integrated in their system, increasing their interactiveness by engaging with the user in a hassle-free conversation.

From a historical perspective, this technology has been explored since the development of ELIZA, a system created by Weizenbaum in 1966 that had the objective of simulating a Rogerian[1] psychotherapist in the most convincing way possible [93]. One of the main motivations of chatbot developers is to develop a software that is capable of passing the Turing's Imitation Game [94], which incentivized the creation of the Loebner Prize Competition in 1991 by Dr. Hugh Loebner.

Conversational agents can be classified according to several factors: interaction mode, final application, knowledge domain and the type of response design approach [95]. Usually, commercially available chatbots are programmed to work in a specific closed domain and to accomplish a given task (task-oriented chatbots), for which the conversation is short and focused on the problem. On the other hand, non-task-oriented chatbots are normally used for entertainment purposes or in applications that require a higher level of engagement between the user and the agent, since the conversations are more natural and complex, often in open domains. The different classifications can be seen in Figure 3.1.

[1] Person-centered

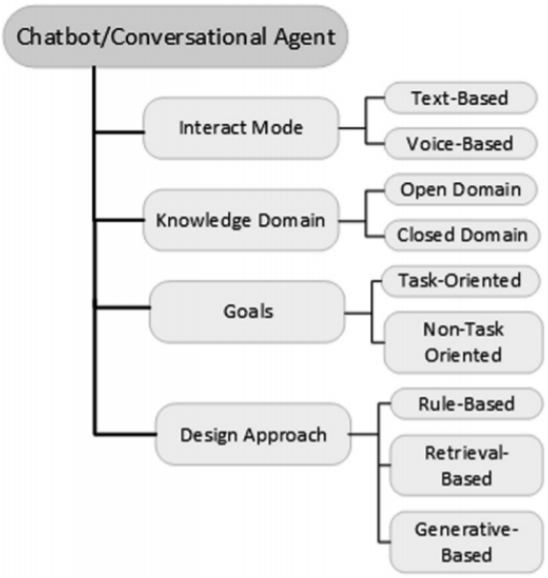


Figure 3.1: Classification parameters of chatbots adapted from [95]

More often than not, chatter bots follow the architecture of Figure 3.2 [80], [96]. As it is possible to observe, there are some essential modules that the system must have in order to fully process user input and provide the adequate response.

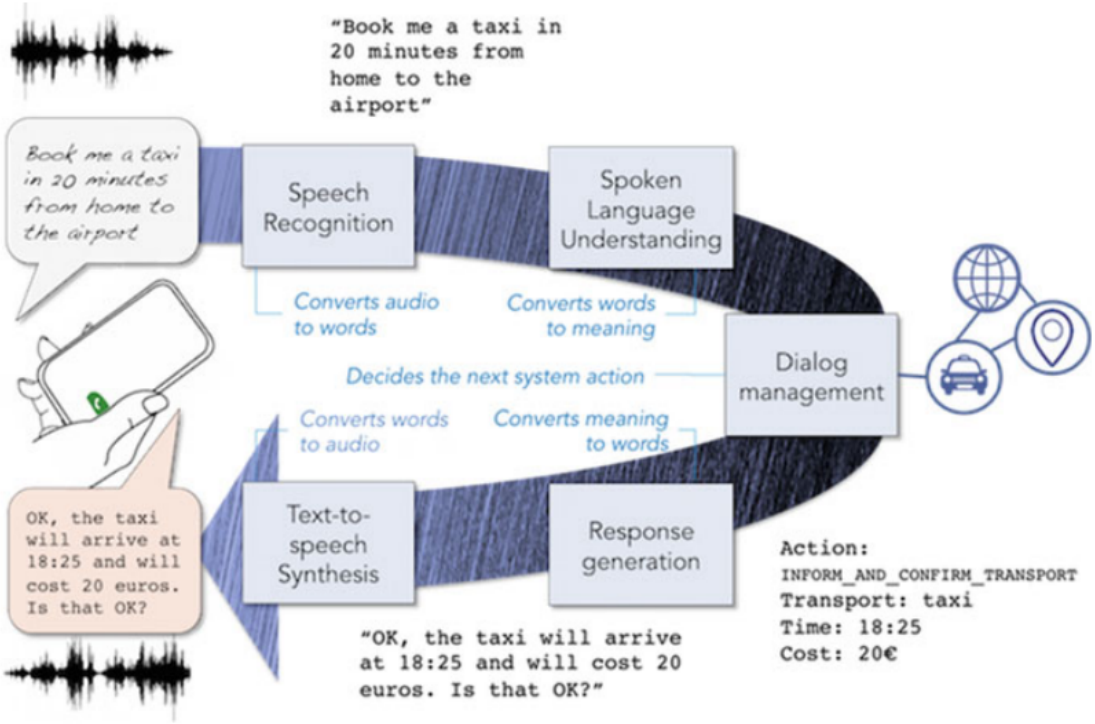


Figure 3.2: Usual chatter bot architecture. The system comprises several modules: speech recognition (speech-to-text), language understanding, dialogue manager, textual response generator and a text-to-speech synthesizer [80]

In the first place, the user voice is captured by the Automatic Speech Recognition (ASR) module which converts the sound in a text input. The textual data must then be comprehended by the computer. This task is performed by the Natural Language Understanding module that makes use of Natural Language Processing and Information Extraction techniques to transform the data into a format that can be used by the machine. Additionally, the Dialogue Manager and the Response Generator work together to select the action that best fits the user input and create the respective answer in a text form. The final stage corresponds to the Text-to-Speech module that processes the generated response in order to synthesise the respective voice sound [80]. The previously mentioned modules will be explored in the following sections.

3.1.1 Speech Recognition and Synthesis

The experience of using a conversational interface becomes complete and more immersive when the main way of communication is speech rather than text. For this, the system must be able to understand the user voice and produce spoken answers [80]. The two processes referred above are known as Automatic Speech Recognition and Text-to-Speech.

3.1.1.1 Speech Recognition

Mobile digital personal assistants like Apple's Siri[2], Google Assistant[3] or Samsung's Bixby[4] were the main propellers of speech-based applications and ASR evolution. These systems have greatly improved recognition accuracy which led to a massive adoption. In fact, according to a study performed by the Pew Research Institute, nearly 46% of American people use digital voice assistants in their daily life. These systems have the final objective of recognising the speech independently of the speaker or the used vocabulary. Bearing this in mind, several dimensions of these models should be taken into account.

- **Vocabulary Size:** The number of words that an ASR system is able to recognise. The current systems have vocabularies of millions of words;
- **Speaker Independence:** These systems should be able to recognise any type of speaker. However, strong accents, speech disabilities or irregular speech patterns can be a challenge for the model;
- **Coarticulation:** In normal flow speech, people tend to make continuous sounds even when changing from word to word. The combination of the ending sound of one word with the initial sound of the following is a challenge for ASR systems, since the boundaries between words are not clear;
- **Conversational speech:** Spontaneous speech like the one verified during conversations, which do not have a predefined structure is harder to process than read speech. In fact, conversational speech is highly prone to hesitations and speech mistakes. These factors

[2]https://www.apple.com/siri/
[3]https://assistant.google.com/
[4]https://www.samsung.com/pt/apps/bixby/

pose a real challenge not only to the recognition systems but also to the following language understanding modules;

- **Robustness against noise:** The surrounding conditions can represent a difficulty to ASR systems. Background noise can be stactic like music or speech or temporary like traffic movement. Signal processing methods like filtering and signal separation are very important to make speech recognition systems more robust to unideal conditions. Beyond that, the exploration of deep learning techniques in this area is known to bring valuable developments for the acoustic models;

- **Microphone:** The distance from the user to the microphone is another variable that must be taken into account. Microphones should be able to capture high quality sound both from a close distance between mouth and microphone and a situation where the user speaks from across a room. In this sense, far-field speech recognition improvements have been made, using microphone arrays and beam-forming technology that allows for larger distances and a direction independent interaction.

ASR is performed as a stochastic process that is based on the probability of a given input matching the word and sound models of the considered language. The main reasons behind this probabilistic approach are the high level of variability and the vagueness of the input data [97]. Speech suffers from inter and intra-speaker variation as well as noisy conditions and variations in the signal acquisition (different microphones and processing units). Therefore, the pronunciation of a given word is never the same and there is not a direct match between the acoustic representation of the input X and a word or phrase [80]. So, the model calculates the matching probability for every word or phrase W in the vocabulary of a language L. This process can be described by the following equation (Equation 3.1) of conditional probability, where $\hat{W}$ is the chosen output:

$$\hat{W} = \arg\max_{W \in L} P(W|X) \tag{3.1}$$

In other words, these systems calculate the probability of the output W given the acoustic input X. Although, this is not possible to calculate in a direct way. The Bayes' theorem is used to solve this issue resulting in the following equation (Equation 3.2). The equation can be further simplified since the denominator takes always the same value and hence the goal is to obtain the maximum, it can be removed.

$$\hat{W} = \arg\max_{W \in L} P(X|W)P(W) \tag{3.2}$$

The term P(X | W) represents the acoustic model, while P(W) corresponds to the language model, which are the models used to fully recognise voice input and will be briefly explained below.

- **Acoustic Model:** This model gives the probability that an output X is given by a certain group of words W. Moreover, the model is trained using a corpus annotated with phonetic

information. The sound that is produced by the user is captured as an acoustic and continuous signal and is then converted to a digital discrete signal. This signal is processed in order to produce an acoustic representation, based on extracted features. Then, the features are matched with the phones (acoustic units) from the acoustic model. After this step the matched phones are grouped to produce the words predicated on a pronunciation dictionary. The generated word sequences are then evaluated to obtain the one that is more probable to happen under the Language model. For this process, the traditional method corresponds to the application of Hidden Markov models combined with Gaussian mixture models (GMM). However, deep learning mechanisms have been replacing these algorithms in the modelling of speech variation [98], giving better results than GMM models [99], [100].

- **Language Model:** As previously stated, the Language model consists in the knowledge base of the considered language. It comprehends the acceptable groups of sequential words and also the likelihood of a given word being part of a sequence. These models can be handcrafted by coding all the grammatical rules which gives the possible sequences or statistically inferred by analysing n-grams and extracting patterns. As it would be expected, grammar based models work better in closed domain contexts where the possible interactions are limited. In addition, these models present less flexibility and robustness when being confronted with word sequences that were not considered at the time of creation. On the other hand, n-gram models can be used for large vocabulary applications although they require a large corpora of spoken data. This method is based on the assumption that it is possible to predict a word in a sequence if the previous $n-1$ words are known.

3.1.1.2 Speech Synthesis

Text-to-Speech (TTS) consists in generating a voice output from a text input. Speech quality and comprehensibility are crucial in the adoption of speech-based applications. There are many implementation of TTS modules where the system must be able to generate the sound instead of reproducing prerecorded audio of the responses. This operation is comprised by two parts: text analysis and waveform synthesis. The usual steps involved in TTS are represented in the Figure 3.3 and will be further explored.

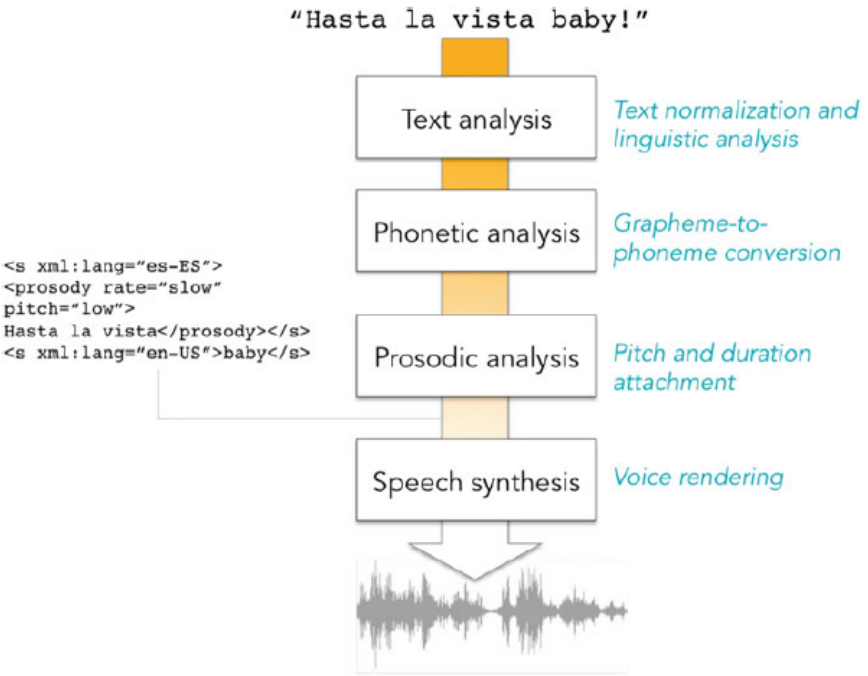


Figure 3.3: Comprehended steps in the Text-to-Speech process: Text analysis, prosodic and phonetic analysis and Speech synthesis [80]

- Text Analysis: The text that is generated by the Response Generator module is firstly analysed and processed before being outputted as voice. A normalisation method is applied to the text in order to prepare it for speech synthesis. As a matter of fact the generated text may have issues in the punctuation and sentence boundary. If this is not solved before the sound is produced, the words will be spoken in a continuously without breaks, making it more difficult to understand. Also, abbreviations and acronyms , as well as, numbers, time, dates or currencies must be changed into a format that can be spoken. Sentence boundaries are determined through tokenization, though the full stop not always corresponds to the end of a sentence. It may also be used in abbreviations like Mr. or dates such as "03.01.1997". Machine learning classifiers are used to classify the punctuation marks as being part of a word or as a sentence boundary. Another related problem are the homographs such as the word "live" can be pronounced in two different ways, having different meanings. This challenge can sometimes be overcome with part-of-speech tagging.

  After normalisation, it is relevant to analyse the phonetics and the prosody of the sentence.

  - Phonetics deals with the pronunciation of the words in the generated text and often uses dictionaries for it. When the pronunciation is not defined in a pronunciation dictionary, a process called grapheme-to-phoneme conversion takes place. This operation uses predefined rules to transform spellings into pronunciations. Also, in the case of homographs with the same part-of-speech a markup language can be used to differentiate the cases (eg. Speech Synthesis Markup Language SSML[5])

[5]https://www.w3.org/TR/speech-synthesis/

  - Prosodic features are the ones related to functions such as pitch, rhythm, loudness, phrasing and tempo, also known as suprasegmental features. A clear example of the importance of prosody in speech is the difference of intonation between a question and an affirmation, which defines the action of the considered utteration. Additionally, prosody have an important role in the disclosing of emotions and in giving synthesised voice a more natural feeling. There are three main elements of prosody: (1) Prosodic phrasing: Deals with the way words are grouped together. Punctuation is crucial in phrasing by indicating, for example, the pause moments between words; (2) Prosodic prominence: the words in a sentence do not all have the same importance. Therefore, some words may need to be highlighted through accentuation, longer duration or emphasis; and (3) Tune: as referred before, the intonation is used to create question sounding sentences (rise at the end of the sentence) or the feeling of admiration. These features can also be modelled using SSML.

- Waveform Synthesis: The first and immediate approach to TTS is the concatenation of prerecorded speech segments that are stored in a database. The segments can go from phones to complete sentences. These are selected according to the best fit to the representation of the considered text. The selected units are joined and signal processing techniques are applied to smooth the junctions of the units in order to increase the naturalness of the speech. Also, there are other methods based on biological mimicry and articulation, however the firstly mentioned method produces better results than these methods when considering the natural feeling and understandability of the speech. The disadvantages of using recorded speech are the specialised human resources that are necessary to record the possible combinations and the low flexibility when dealing with open domain applications or unseen contexts. More recently, the work of van den Oord et al. [101] in the DeepMind[6] research group have showed progresses on speech generation using deep learning algorithms. The method uses WaveNet, a Convolutional Neural Network that takes in to account the context of the input and therefore produces very natural results as stated in the study, whose results are shown in Figure 3.4.

[6]https://deepmind.com/

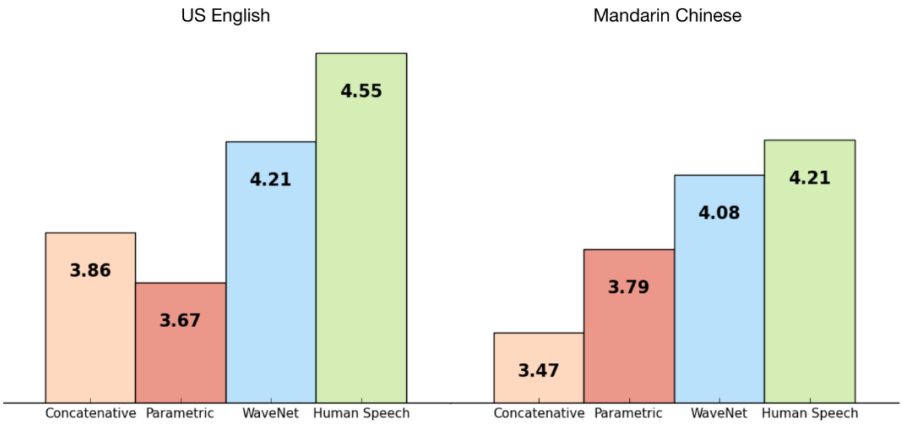


Figure 3.4: Comparison between the DeepMind's Text-to-Speech approach (WaveNet) and other traditional approaches, regarding Human preference for both US English and Mandarin Chinese. The results were obtained using Mean Opinion Scores (from 1 to 5), which are the standard measure for subjective sound quality tests. The opinions came from a blind test in human subjects, from more than 500 ratings on 100 test sentences, adapted from [102].

3.1.2 Communication as an action

Wittgenstein proposed the idea that people execute actions with the utterances produced during a dialog [103]. This work led to the theory of speech acts that was a further development of the aforementioned work by Austin [104] and Searle [105]. This theory states that the performance of a speech act needs the fulfilment of some conditions. So that an utterance is intended as a command by the speaker and received as one by the addressee, the presented requirements must be fulfilled [105]: (1) the utterance is related to a future action that the hearer should execute; (2) the act can be done by the hearer and the speaker believes in that; (3) the execution of the act by the addressee, in the normal course of events is not obvious for the speaker, nor the hearer; (4) the speaker intends the hearer to realise the action. These conditions summarise the intuitions that people have when asking to perform a given task that they want to have done and that they suppose that the hearer can do, and otherwise would not do so if not asked to. Understanding the speech act that is being performed when using an utterance is a key procedure in order to determine the speaker's intention and consequently answer in the correct way [80], [106].

This task is a key topic when dealing with computational systems of conversational agents because the users do not always express the intentions they have in a direct way. Due to this, these models should employ mechanisms that allow them to infer user intentions using the provided information. An illustrative example of this happens when the user says that he/she does not like one ingredient after receiving a recipe recommendation. In fact, the primary user intention is to have the given ingredient replaced by another. The system must acknowledge that and give an appropriate answer.

3.1.3 Intent Determination

Speech act recognition consists in the task of defining the function of an utterance in a dialog, for instance, if it is a suggestion, a question, an offer, or an acknowledgement, among others. This task used to be performed applying early AI models of plan inference and reasoning [107]. This way, logical rules were applied to the knowledge base so that the new information is deduced. As an alternative, statistical/machine learning models appeared afterwards and deal with speech act determination as a supervised classification problem. Thus, these last models require a training phase using a labelled corpus[7] containing speech act annotations in each utterance [80], [108]. The corpus should contain domain-specific information so that the model is optimised for those utterances. As was to be expected, the models use textual features to classify the information. Among the features used it is possible to highlight: (1) key words and phrases, as well as punctuation marks present in the utterance [109], [110] (lexical information), normally as n-grams[8] like "please", which is a strong indicator of a request; (2) syntactic and semantic data from linguistic relation processing [108]; (3) preceding dialog acts and its n-grams [111] and also sequences of dialog acts in the form of n-grams [112] as contextual features and also (4) prosodic and acoustic features in the case speech recognition modules are implemented [113].

The aforementioned functions (speech acts) can be combined with domain knowledge or tasks in order to characterise and define the user's *intent* [80]. User *intent* is a key concept for chatbot construction. It represents the intention that the user wants to express. The complete classification of the user's intent requires the detection of the appropriate relevant elements. According to Wu et al. [114], the two step classification helps to constrain the utterances content semantic processing, since the identified elements must then be related to the previously identified intent. In order to extract the abovementioned information the techniques explained in Section 2.3 are applied. Besides these techniques, the methods explored in the following sections are also used. These techniques have the advantage of being an unsupervised way of generating textual features.

3.1.3.1 Bag of Words

The user content can be represented in many formats. One of them is the Bag of Words (BoW) which collects only the words in the input, excluding the syntactical properties. Moreover, stop words are often deleted and lemmatised word occurrences are counted [84]. This way, the final representation is a vector containing the frequency of each word in a text sample. This approach is regularly used in word matching processes like document retrieval or question answering applications. The Bag of Words approach has the advantage of being simple since it does not require linguist knowledge. However, some complex tasks require methods that take into account a larger amount of information, for example semantics and syntactics [80]. In the work of Ni et al. [115], a chatbot aimed at providing primary care assistance in patient intake is presented. The patients are able to expose their symptoms by describing them to a conversational agent. Furthermore, the implementation of the natural language understanding engine relies on a bag-of-words methodology to group all the symptoms of the patient. This way, the authors

[7] structured set of texts from a given language used to perform linguistic statistical analysis and hypothesis testing
[8] sequential group of textual items, such as words, present in a text sample (corpus)

can overcome the obstacle of lay language usage and information sparsity when extracting the important elements of the input. The algorithm uses the information of the bag-of-words and compares it with the symptoms of diseases in the database and produces a list of the most probable diseases related to those symptoms.

3.1.3.2 Word Embeddings

As referred in section 2.3.3, word embeddings can also be used to classify the intent of a sentence. In this case, the word vectors will be used as features for the classification models. Besides the advantage of considering word semantics, this method also produces smaller and non-sparse vectors, opposite to Bag of Words. When using deep learning algorithms, this can be very important since it reduces the number of weights needed and consequently the computational cost. In this case, it is possible to consider both static and contextual embeddings for this purpose.

In order to increase performance, the word vectors should be related to the domain. Ideally, when using static word embeddings, they should be obtained using a corpus that captures the relations of the domain. This requires a huge set of data that sometimes may not be available. The same applies to pre-trained language models. These models can be fine-tuned to increase performance in domain knowledge. It would be expected for the results produced by contextual embeddings to be better than the ones obtained using static embeddings. However, in fairly closed domains, this difference may not be noticeable, if there is not a high level of ambiguity. Nevertheless, methods based on embeddings allow for a better performance than Bag of Words. This technique is frequently used in cases where the models rely mainly in numeric representations of the text data such as deep learning approaches. As a matter of fact, the work developed by Xu et al. employs a word embedding algorithm to enable the utilisation of Recurrent Neural Networks, more precisely Long Short-Term Memory units [42]. In this case, the authors utilise a word2vec neural network language model based on conversations from the domain that they are trying to tackle (customer service). The work developed by Dundar E. et al applied an hybrid approach that combined keyword with word embeddings for a question-answering chatbot [116]. They used the embeddings as an extension for the keywords matching. The embeddings were obtained using a corpus of domain related conversations that was available.

3.1.4 Dialog Management and Response Generation

Chatter bots use the information extracted using some of the methods explored above to generate the answers to the user input.

3.1.4.1 Rule-based

The great majority of chatbots are rule-based, meaning that the answers provided are generated by a set of rules which are normally hand coded. The answers are selected from a predefined knowledge base according to the user input, hence it does not create new text. Thereby, the construction of this knowledge base is very expensive in terms of time and human resources, also making it not scalable [117]. This works by matching a received pattern with a template from

the knowledge base and is mostly applied in closed domain applications. The work conducted by Kasinathan et al. [118] employs a rule-based methodology in a chatterbot meant to detect and diagnose illnesses among patients. The system makes use of a knowledge base and a group of inference rules mainly based on IF and THEN statements that allows for response generation. Nevertheless, as was stated above, the system must recognise certain keywords in order to proceed and if a keyword is not understood it will ask for a different input for the same question.

3.1.4.2 Information Retrieval based

However, with the emergence of social networks like Twitter and Reddit that allow the users to establish a conversation, large dialogue datasets and Information Retrieval models were created [119]. Developers could analyse these conversational datasets and automatically create pairs comprised of an utterance and a response. These models receive the input information and pattern match it with the set of utterance and response pairs to select a response. The possible responses are ranked according to the matching algorithm and the higher ranked one is chosen as the output [120]. The disadvantage of using retrieval models is that it can be hard to collect a corpus of the given pairs for some specific domains. Once again, this model do not create new text, it only selects pre-defined responses from a corpus. Lommatzsch et al. [121] explore an information retrieval methodology to build a chatbot that helps users find information in large knowledge bases. This study leans on the premise that the lists of Frequently Asked Questions can serve as a knowledge base for generating the responses that the user requires. Moreover, the match between the user input and the set of texts that has the answer may be made using an inverted index method. Namely, the TF-IDF (Term Frequency – Inverse Document Frequency) algorithm is widely applied. This algorithm corresponds to a numeric statistic that tries to identify the relative importance of each word in corpus. It considers the frequency that each word appears in a document and relates it to the frequency of documents that contain that specific term, weighting the words according to that. Words that high weights are terms that are frequent in a given document but that do not appear in other documents. This way, it filters highly common words (eg. stop-words) or words that appeared very few times and may not be relevant for the domain. To increase the quality of the conversation, the authors also implemented a memory state that considered the context of the inputs, although it takes a manual approach instead of automatically inferring the end of the context. It asks the user whether or not the question that was asked is still about the same topic that was previously defined, in order to give coherent answers. Furthermore, techniques based on word embeddings and language models can be used to select the appropriate response. By encoding of each utterance, the distance metrics or classification models can be used to match user inputs with the several different answers that the chatbot can reproduce. The work developed by Henderson et al. [122] and Gu et al. [123] used pre-trained language models to improve the performance of response retrieval. These models are more aware of the complete conversation context, which can be crucial in multi-turn conversations.

3.1.4.3 Generative models

On the other hand, and as the name implies, Generative models create text output based on the user input. It does so by using an encoder-decoder architecture that firstly creates a vector representation of the user's utterance which is then decoded to generate the answer. As it would be expected these models need an annotated training corpus. This is often seen as a translation problem in which a query is translated into a response [119]. Moreover, Generative models can make use of different algorithms. Besides the more traditional statistical based machine learning algorithms, deep learning [124] and reinforcement learning [125] can also be used. These models produce better quality conversations, considered to be more interactive, informative and coherent than the ones generated by statistical models [126]. Moreover, the work of Xu et al. referred in Section 3.1.3.2 [42] employs a deep learning approach based on a Recurrent Neural Network architecture. This model is able to map the user input to an output using an encoder-decoder methodology. The encoder maps the input into a fixed size vector and the decoder maps that vector back to a text format. Both the encoder and the decoder are composed by LSTM units which can keep and learn a memory state that is fundamental to filter sequential information and generate more comprehensible outputs. This technique is called sequence-to-sequence (Seq2Seq) learning, in which a sequence of input words is mapped into a sequence of output words. The system was built to assist in customer services, hence it was trained on a corpus of *Twitter* conversations. The dataset was composed by approximately 1 million conversations between users and agents from several brands. Once again, language models may be applied in this task. The work conducted by Wang et al. [127] concluded that BERT can be used to sample sentences. The results obtained by these authors showed that this technique can produce a wider range of sentences that the Seq2Seq algorithms, although the quality is inferior.

3.1.5 Knowledge bases

As seen previously, chatbots need information bases in order to generate appropriate responses to user intents. The information can be coded or represented in several ways. Additionally, in task-oriented chatbots it is important to have a service to perform the jobs that are required, such as a call to an API.

3.1.5.1 Relational Data Bases

One of the most commonly used knowledge base type are Relational Data Bases (RDB). This technique can be used to store domain information or dialogue data, so that the bot can access and use previous inputs and responses. This way, it can make the conversation more coherent and meaningful. RDB's are usually implemented using SQL (Structured Query Language) language, meaning that the information retrieved from the user input must be converted into a SQL query, and vice versa, so that the system is able to insert and retrieve information from the data base.

3.1.5.2 Ontologies

Another way to provide the system with world knowledge is to use ontologies. An ontology (or semantic network) consists in a set of relationally and hierarchically connected concepts. It is a logically structured representation of the relations between concepts of the domain, that can be graphically represented [128]. Ontologies are often build with taxonomic hierarchies of classes, comprising class definition and the encompassed relation [129]. This way, intelligent agents can perceive the given relations and even imply new statements using reasoning rules [130]. It is also important to standardise and systematise the model that is being created, so that information retrieval becomes simpler and more organised and based on a defined vocabulary [129]. Therefore, ontologies define the conceptual universe of knowledge of the system, grouping all the important information in one interconnected framework. Once again, the extracted user intent format must match the ontology one so that it is possible to retrieve the desired information.

3.2 Chatbot construction tools

In this section the tools for chatbot development will be explored. The inner workings, as well as advantages, disadvantages and possible challenges for the upcoming work.

3.2.1 Speech Recognition and Speech Synthesis

Integrated frameworks that comprehend both Speech-to-Text and Text-to-Speech modules are widely spread and used in chatbot construction. Probably the most used one is Google Cloud Speech-to-Text[9] and Text-to-Speech[10], although other commercial solutions are available. The following section explores some of the most used frameworks for the considered tasks. It is advantageous to use frameworks that have both speech recognition and synthesis since the integration is simpler and seamless. A complete comparison between both Speech-to-Text and Text-to-Speech tools is present in Tables 3.1 and 3.2, respectively.

3.2.1.1 Google Cloud

Google Cloud speech-to-text module is capable of processing a large vocabulary in 120 different languages. It employs deep neural networks for speech recognition and is able to perform real-time text generation. Noise reduction and content filtering are used to increase robustness and achieve better quality transcriptions. Furthermore, there is the possibility of customisation on both the model selected (different optimisation processes according to the context, for example phone calls, voice input or video) and some keywords of the application domain. There are also some beta features that can be implemented to elevate the interactiveness and quality of the agent: automatic punctuation and language detection and speaker diarisation for multi-speaker contexts.

[9] https://cloud.google.com/speech-to-text
[10] https://cloud.google.com/text-to-speech

It is also important to consider the pricing associated with this service. It has a free plan that is very advantageous for research purposes since the necessary features are included.

Google Cloud services has also a Text-to-Speech module that is freely available for every user. As with the Speech-to-Text function, Google's TTS uses deep learning approaches in order to generate voice outputs that resemble human interaction. As a matter of fact, this framework uses the WaveNet work developed by van den Oord et al. explained in Section 3.1.1.2, thus making it a generative based method for speech synthesis. It also supports 30 different idioms, including Portuguese and English, speech velocity adjustments and SSML for prosodic and phonetic features.

3.2.1.2 Microsoft Azure

Another widely used technology for speech recognition and speech synthesis are the services provided by *Microsoft Azure*[11]: Bing Speech-to-Text and Text-to-Speech. This service uses the stochastic approach described in Section 3.1.1.1 for the speech recognition, while creating the models through recurrent neural networks architectures [131]. Moreover, it allows for customisation of the Acoustic Model, the Language Model and the accent in several languages, such as English and Brazilian Portuguese. However, for European Portuguese these customisation options are only available for the Language Model. On the other hand, the Text-to-Speech module has two different approaches to speech synthesis. The Neural Network based approach, which gives better results by taking into account prosodic features and smoother word transitions, is only available in a few languages, including English. It performs prosody prediction and speech synthesis simultaneously, allowing for a more natural speech and reduced listening fatigue. The second approach, that uses pre-recorded sounds and processes the prosody and the phonetics in two steps, has compatibility for European Portuguese. That being said, it is also worth highlighting the training option that this service provides. In fact, it is possible to train the implemented software by giving annotated data. This allows for improved accuracy in specific domain contexts, regarding speech recognition. When considering speech synthesis, there is the possibility of creating a custom voice using speech records and both plain text or SSML. It also has a free instance that has a limited time (ASR) or number of characters (TTS).

[11] https://azure.microsoft.com/en-us/services/cognitive-services/speech-services/

Table 3.1: Speech-to-Text tools comparison

	Google Cloud STT	Amazon Transcribe	Microsoft Azure STT
Technology	Deep RNNs	Deep RNNs	Deep RNNs
Audio Stream	Yes	Yes	Yes
Customisation	Model(phone call, video or voice) 5000 domain related keywords	Domain related keywords	Accent, Language Model, Acoustic Model, Training with own data
Languages	120	6	40
EU Portuguese	Yes	No	Yes
Extra features	Automatic punctuation; Language Detection; Speaker diarisation; Noise reduction and content filtering	Automatic punctuation Speaker diarisation	Automatic punctuation; Speaker diarisation; Noise reduction; Sentiment Analysis
Pricing	Free tier	Free tier	Free tier

Table 3.2: Text-to-Speech tools comparison

	Google Cloud TTS	Amazon Polly	Microsoft Azure TTS
Technology	Standard: Statistical Parametric and/or Concatenation Synthesis; Neural: DeepMind's WaveNet	Standard:Statistical Parametric and/or Concatenation Synthesis; Neural: Deep Learning	Standard:Statistical Parametric and/or Concatenation Synthesis; Neural: Deep Learning
Customisation	Speech rate and audio profile	Speech style (newscast or conversation); Speech rate; Own voice (external service)*	Creation of model using own voice
Languages	Standard: 30 Neural: 20	Standard: 29 Neural: 4	Standard: 45 Neural: 5
EU Portuguese	Yes (Both)	Yes (Standard only)	Yes (Standard only)
Extra features	SSML; Prosody prediction (neural voice); 180 different voices	SSML; Change the lexicon/vocabulary (change pronunciation of specific words)	SSML; Prosody prediction (neural voice)
Pricing	Free tier	Free tier	Free tier

* Amazon provides this service by request

3.2.2 Intent Identification and Dialogue Management frameworks

As with the speech services, frameworks for intent recognition and dialogue management are commercially available. These tools apply NLP algorithms automatically in order to create the Natural Language Understanding engine that controls the flow of the conversation. The great majority of these frameworks are based on the intent concept and its identification, being widely used for task-oriented chatbots [121]. In order to explain intent-based approaches, the case of Dialogflow will be explained. The alternative to the use of these frameworks is the development

and integration of the chatbot using NLP tools and training corpora that can be a public dataset or a domain-specific corpus that must be collected and annotated by the team of developers. The comparison between the several frameworks is represented in Table 3.3.

3.2.2.1 Dialogflow

Dialogflow[12] is Google's dedicated tool for the end-to-end creation of conversational agents. It runs on the Google Cloud Platform, combining Google's expertise, products and extensive machine learning resources to allow the integration in several applications. This service is used by several companies that already have market products to take consumer experience to the next level. Moreover, Dialogflow is also used in research and development for academic purposes. It can deal with several different languages, including English and Portuguese.

The service provided by Google encompasses several important objects that will be further explained.

- **Agents**

  In Dialogflow, agents are the objects that handle the conversation with the end-user. An agent is comprised by a natural language understanding module that has the objective of comprehending the human language and extract structured information that may be used by other services. In other words, the agent is similar to a call centre employee that is trained to deal with predictable scenarios with a high level of flexibility [132];

- **Intents**

  As previously stated, an intent defines the perceived intention of a user when communicating with the conversational agent. Each agent requires the definition of the possible user intents. The user expression (utterance) is matched with the best intent that the agent is able to detect. This matching process is also called intent classification [133]. In the example given in the Dialogflow documentation, a "weather" agent is created and the "forecast" intent is defined. This intent will match expressions that require weather forecast information, as seen in Figure 3.5. Moreover, the agent can also be coded to extract required information from user expressions that will be necessary to compute a given query. In this case, the desired time and place are important data to give a more specific response.

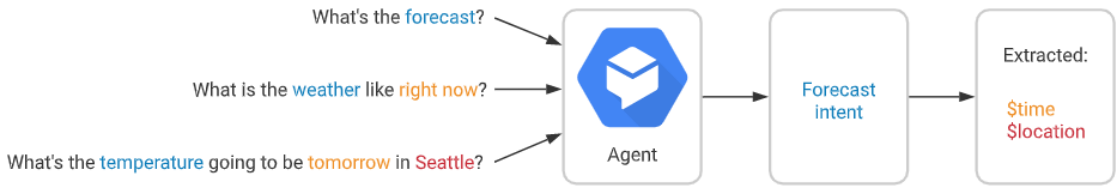


Figure 3.5: Dialogflow intent matching example [133]

The intent flow in Dialogflow is represented in Figure 3.6 and comprehends the following blocks [133]:

[12]https://dialogflow.com/

- Training phrases: When creating an intent, training phrases should be provided to the system so that the agent learns what the end-user may say when requiring a specific action. The training phrases constitute the corpus of each intent, which is further expanded by the natural language processing algorithms of Dialogflow. This means that not every possible combination of phrases that correspond to a given intent must be inserted, since the system is able to understand the underlying meaning of the action and to create similar sentences to train.
- Action: As expected, the intents are related to actions that can be triggered after intent classification. It is possible to define the action that each intent triggers.
- Parameters: When the system matches a user expression with an intent it is possible to extract the data in the expression as parameters. These parameters can later be matched with entity types used to generate responses or initiate any other action in the system. The data comes in a structured way, which facilitates the mentioned functions.
- Responses: The answers that the system produces can also be defined in the intent. These responses might be textual, speech or visual in order to present some information, ask for more data or end the conversation.

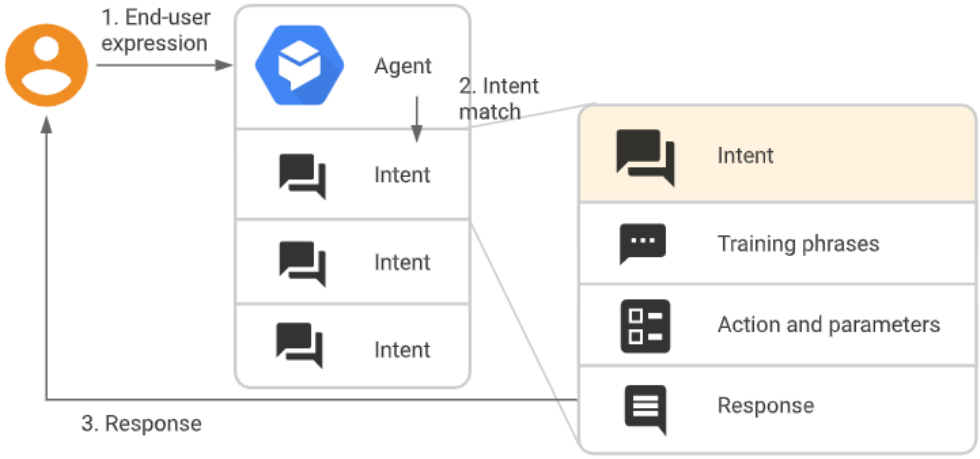


Figure 3.6: Dialogflow intent matching and response flow [133]

- **Entities**

  The parameters referred in the previous item are called entities and correspond to the entities mentioned in Section 2.3. An entity type is attributed to each entity after guiding its extraction from the user expression. Dialogflow has several entity types already defined that can be used by the developer. Besides these, there is the possibility of creating new entity types by providing examples to the system [134]. Under the scope of this work, it would be necessary to create, for example, an entity type for food items to match the ones that are present in the recipes.

- **Context**

The conversational agent often needs context so that the generated response addresses the objective of the interaction. Therefore, the conversation flow requires context in order to become more natural and comprehensible. The intent configuration allows for the definition of contextual inputs and outputs and this is used by the agent to keep coherence along the conversation [135].

- **Follow-up intents**

  It is also possible to determine combinations of intents that usually happen sequentially. The follow-up intent is implemented as a child of a given parent intent and will only be matched if the latter is matched in the previous turn of conversation. These pairs of intents are joined by a generated context that relates both. In the parent intent an output context is created, while in the child intent, a matching input context is generated. This feature is not limited to two intents, as it is possible to create nested structures of multiple follow-up intents [136]. The example provided in Figure 3.7 shows how the pairs of intents can work to make a special type of appointment, namely a haircut appointment. This is particularly useful in scenarios of "yes" or "no" expressions.

Intent name	Training phrase	Input context	Output context	Intent response
Appointment	Hello		appointment-followup	Would you like to make an appointment?
↳ Appointment - yes	Yes	appointment-followup	appointment-yes-followup	Would you like a haircut?
↳ Haircut - yes	Yes	appointment-yes-followup		Your appointment is set.
↳ Haircut - no	No	appointment-yes-followup		Goodbye.
↳ Appointment - no	No	appointment-followup		Goodbye.

Figure 3.7: Dialogflow follow-up intent example with a haircut appointment. The intents are combined by the input and output contexts [136].

Bearing in mind that the developed chatter bot is supposed to interact with the recommender system, the connection between Dialogflow and the system should be processed in the following way [137], represented in Figure 3.8:

1. User input;
2. The system sends the user expression to the chatbot API so that the intent and further data is detected and extracted;
3. After intent detection and matching, the response (with the required data – intent, action and parameters) is generated and is sent back to the system;
4. Using the information in the answer, the system performs the necessary queries and other operations;
5. The final response is then sent to the user;
6. The user is able to read or hear the response and obtain the desired output.

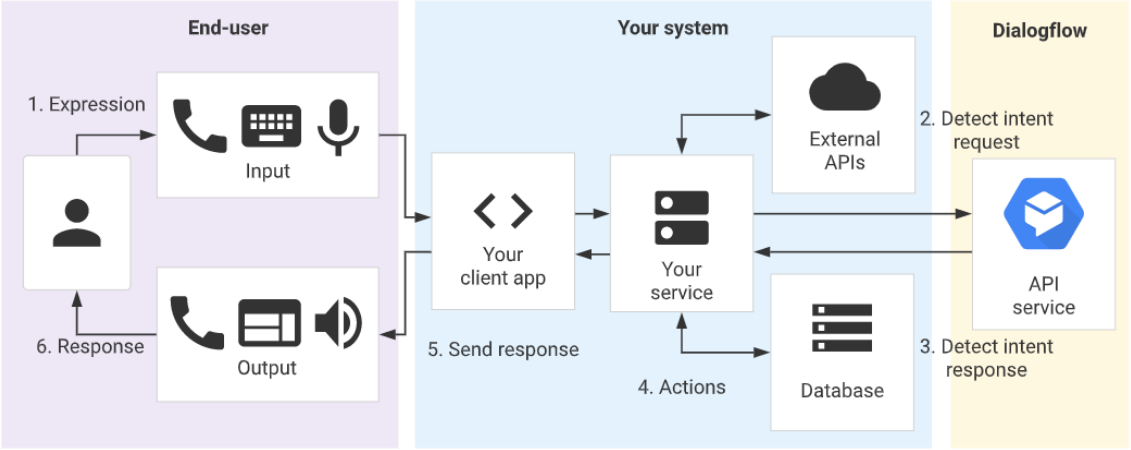


Figure 3.8: Interaction between a system and the Dialogflow API. It is processed according to the below described steps [137].

3.2.2.2 Wit.ai

Wit.ai[13], another chatbot construction tool, follows a similar approach to the one described in the previous section. The major difference relies on the fact that Wit.ai is not able to generate responses, it focus only on intent classification and entity extraction to return a structured JSON file with the information. This information can then be used to query a database or trigger rule-based responses in an external module. Moreover, this software is open source and completely free. Therefore it is possible to access the code of the software and change it according to the user will.

3.2.2.3 LUIS.ai

Microsoft's natural language understanding unit is called *LUIS.ai*[14]. This engine is very similar both to Dialogflow and Wit.ai (intent-based). It is used to retrieve user utterances in structured JSON files that can be used to query knowledge bases. However, the dialog manager component is hard to customise. In fact, LUIS.ai has to be integrated with the Azure Bot framework so that it is able to respond to the user. It uses pre-stored responses instead of generated ones, which decreases the naturalness and immersion of the conversation. Some advantages that are present in Microsoft Azure services are the integration with both the speech interface referred in Section 3.2.1.2 and a database hosting service. It also has a free option for developers.

3.2.2.4 Rasa Open Source

Another player in the chatbot construction domain is Rasa, an open source framework that has become increasingly popular due to its flexibility and also to the fact that it is open source and free. Rasa uses an approach similar to Dialogflow – an intent and entity-based agent – and includes also the part of dialogue management, opposite to Wit.ai. Being open source,

[13]https://wit.ai/
[14]https://www.luis.ai/home

this framework grants a huge level of customisation of every single aspect of the process. It is possible to customise the classification models used for intent/entity recognition, as well as the model that controls the next actions or responses of the chatbot. The NLU pipeline is also customisable, enabling the addition of specific components, like sentiment analysers or spellchecking modules. The dialogue management component is retrieval based, which can be sufficiently good for closed domain agents, although it demands the creation or collection of possible answers.

Table 3.3: Comparison between several available frameworks for chatbot creation – intent identification and dialog management

	NLU and Dialogue Management	Integration	Graphical Interface	Languages	EU Portuguese	Extra Features	Open Source	Pricing
Google Dialogflow	Active learning: intent classification and generative response creation (10-15 utterances, each)	ASR, TTS, Google Assistant, API for external integration	Yes	32	Yes	Training using own data (conversation logs) Sentiment Analysis* (Text, Sentence and Entity level) Rich responses (other than text) Small Talk Multi-language agent	No	Free tier
Wit.ai	Active learning: intent classification; Does not have response generation	ASR (32 languages, including portuguese), API for external integration	Yes	132	Yes	Sentiment Analysis (Sentence level) - Requires training	Yes	Free
Microsoft LUIS.ai	Active learning: intent classification; Requires Azure Bot Framework for response query	ASR, TTS, API for external integration	Yes	23	Yes	Pattern matching for intent classification Language detection Sentiment Analysis Small Talk	No	Free tier (text input only)
Rasa	Active learning: intent classification and generative response creation	API for external integration	No	Depends on the word vector model used (eg. spacy, MITIE)	--	Customisable modules (eg. sentiment analysis, small talk)	Yes	Free (text input)
Amazon Lex	Active learning: intent classification; Rule-based (user defined) response output	ASR, support for TTS, API for external integration	Yes	English only	No	Sentiment Analysis	No	Free (first year)

*Premium feature

3.3 Chatbot evaluation

The developed chatbot in this work should be validated, prior to deployment, according to the correct and appropriate evaluation methods and parameters. Evaluation is also crucial to guide the construction of the proposed agent and assess the usefulness of features. Regarding the performance of the generated answers, there are some implemented metrics in the literature that can be used and lean on word-overlapping similarity to evaluate the system:

- BLEU [138] – It uses the occurrence of n-grams to calculate performance
- METEOR [139] – Also considers synonyms of the tokenised words
- ROUGE [140] – Meant to address summarisation; it uses sequence of words that do not have to be contiguous to compare the sentences

These metrics were first developed to validate machine translation; however, they can be adapted to response generation evaluation [141]. These methods do not always give the best evaluation due to the intrinsic variability of speech and possible responses. As word-overlapping methods, valid answers with different wording are wrongly classified with a negative score. That being said, methods based on word embeddings have appeared to circumvent the mentioned limitations. These models compare the meaning of the generated responses with the expected ones. Hence, word embeddings metrics produce better evaluations that are able to distinguish baseline agents from more complex state-of-the-art agents [142]. Moreover, retrieval-based metrics such as recall or precision have also been used to evaluate dialogue agents [143], mostly in task-oriented chatbots.

However, the presented metrics do not properly represent the human judgement [144], which comprehends a larger set of dimensions (such as coherence, expectation, conversational knowledge depth, sentimental behaviour, trust, domain coverage and engagement), besides the correctness of the information provided [145]. As a matter of fact, several studies have proved that human opinion on dialog agents can be very different from the assessment provided by these metrics [142], [146]. Therefore, human judgement should be taken into account when validating a conversational agent, although this can be prohibitively expensive in terms of specialised human resources. It is virtually impossible to properly evaluate every iteration of an agent that is being developed with human-based experiences. That being said, machine learning based evaluation models that incorporate human opinions as training data have been developed, such as ADEM [147], a RNN based model that uses context and human opinions to predict judgement on both sentence and system levels) or RUBER [148], which uses word embedding to assess similarity with the ground truth and combines it with the result of a RNN that verifies the relatedness of the query and the response. More recently, methods using adversarial learning have been applied to automatic evaluation of response generation systems, showing that deep learning models can be trained to discriminate computer-generated responses from human responses [149].

Although these automatic evaluation models and metrics try to close the gap to human judgement, human-based experiences should also be conducted for a complete evaluation of the conversational agent. To do so, relevant metrics and ways to quantify them should be defined.

User satisfaction can be assessed on a session level and on a response level. The former deals with the user feeling for the whole session of conversation with the agent, while the latter evaluates the appropriateness of responses. The most common methodology is the utilisation of Mean Opinion Scores, in which each user is asked to grade the specific parameter that is being analysed with a value from a small interval (for example, from 1 to 5) or an equivalent qualitative classification (for example, *excellent*, *good*, *bad*) [150]. Regarding user experience, there are benchmark questionnaires that aim to assess different metrics. Namely, the System Usability Scale [151] (SUS), in which users should express their level of agreement using a symmetric agree-disagree scale (Likert scale [152]) for 10 questions to obtain a numeric usability score out of 100. This test evaluates the system independently of it being a chatbot or not. It is often used for software, websites, hardware or even industrial applications. Furthermore, this metric can be compared to benchmark values in order to understand the overall usability of the system [153]. Another advantage of this system is the strong correlation with the Net Promoting Score. The latter is a metric developed by Reichheld [154] that measures user's loyalty. It is also widely used in Business scenarios to assess the quality of a firm's customer relationships. A similar questionnaire is the User Experience Questionnaire (UEQ) developed by Laugwitz et al. [155]. This one is intended to evaluate interactive systems according to 6 different scales (Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation and Novelty), based on 26 pairs of adjectives with opposite meanings. This questionnaire also has benchmarked results for further comparison and analysis. Field trials can also be designed as A/B tests where the goal is to evaluate two different instances of the same system. The two systems are distributed randomly among the testers for them to evaluate the different versions of the same component. This way it is possible to decide on a specific feature based on human judgement [156].

3.4 Nutrition and Health Related Chatbots

Regarding the actual applications of chatbots in the healthcare field, a number of approaches have been developed in the last 20 years. This section presents some of the related work on nutrition and health related conversational agents.

According to the review conducted by Montenegro et al. [39], counselling and coaching agents are the two modalities that guide the development of health related conversational agents. The latter's function is to train the user to learn a new task or skill. These modules are mainly used for physician training or to create simulated patient environments where students, researchers and physicians can learn and be presented with new cases. On the other hand, counselling agents are used in cases where social engagement is necessary so that the user is encouraged to alter its habits towards healthier ones [157]. Also, they can be used as support tools for patients that suffer from emotional health disorders [158]

As for the goals that the agents are created to achieve, the same report refers that these can be divided in six types: (1) Assistance, (2) Training, (3) Elderly specific, (4) Diagnosis, (5) Education and (6) Prevention. The first type aims to provide help to healthcare specialists and patients in daily tasks. In the case of patients they can be used to assist in therapeutics or treatments that require cognitive exercises [159]. The second type of agents is often used by students or

physicians to learn new skills or to provide hypothetical situations that enable the practice of the attained knowledge. The model developed by Rizzo at al. [160] simulates a patient that allows the training of clinical skills, by means of a conversational agent. The third type focus on the interaction with the elderly population. It is a population that, more often than not, has low inclination to adopt new technologies and try to avoid any unnecessary contact with it. There are some agents created to assist the elderly in the necessary tasks such as health-related or home affairs. The work developed by Tokunaga et al. [161] tried to create a smart virtual elderly care giver. In this work, the author concluded that some of the challenges of developing a virtual care giver for older people are the lack of customisation and learning skills to rapidly adapt to the user needs. The author states that these problems come from the creation of agents that are not specifically fitted to each user and lack the learning skills required to adaptation to the different user contexts and preferences. In order to tackle these limitations the author implement a set of features that are highly customisable and that are able to deal with personal information, thus creating engagement and increasing user trust in the system. The system adapts itself to the user conditions. As an example given by the authors, when dealing with a user that has hearing impairments, the system speaks the responses slower and louder than the default settings. That being said, it is important to be aware of the user impairments and health conditions in order to develop a system that is able to overcome these limitations and provide a superior experience. Moreover, the work conducted by Bickmore et al. [162] employed a conversational agent as a virtual advisor to provide physical activity support and advice in a smart and tailored way, aimed at the elderly population. The agent acted as a counsellor that helped users to change their exercising habits. The authors concluded that the utilisation of a health companion with the goal of motivating habit changes was a successful measure. The results indicated that the group of patients that used the agent increased the level of performed physical activity, when compared to the control group that did not use the companion bot. The fourth group of conversational agents focus its efforts on diagnosis by offering a way to predict diseases based on behaviours or symptoms. These agents can be present either in a clinical context [115], helping a physician to make more correct decisions or in a patient's mobile device [163] where it could track and monitor user behaviour or serve as a journal to register symptoms and their evolution. The last type deals with prevention of conditions or diseases. Under this scope, the agents can provide assistance in the user decision of a prevention pathway, or in cognitive and emotional disorders such as the prediction and prevention of suicidal traits or depressive behaviours [164], [165]. Companion chatbots are usually considered counselling agents that try to change the dietary habits of a user. With this in mind, researchers frequently implement an embodied agent [115], [162], [166] that is believed to increase trust and engagement [167].

More specifically, the work developed by Fadhil [168] studies the challenges related to a chatbot as a means of meal recommendation. The author highlights user engagement as a core concept in applications that deal with lifestyle habits. He states that the humanness of the agent is a crucial characteristic that should be greatly improved. This is made by the generative algorithms involved, both in speech synthesis and response generation. Another important note referred by the author is the personality and coherence in the flow of the dialog. That being said, it becomes very relevant to accurately track the context of the dialog since users expect the agent to remember the previously stated information, such as name, gender or chosen options. A technique that is used in commercially available chatbots is the prediction or recommenda-

tion of the next user request. This way the user is guided to the available tasks and does not feel lost when interacting with the agent for the first time. Moreover, the results obtained by Xu et al. [42] concluded that users' input can be classified in two classes: Emotional requests and Informational requests. In the former, users express their emotions, opinions or attitudes regarding a given topic, while in the latter users require information to solve a given problem. The implemented agent must be able to identify and properly address both requests as well as out-of-domain requests. Furthermore, the survey conducted by Abd-alrazaq [169] reported that rule-based chatbots had a wider adoption in health contexts since this technology is seen as safer and more reliable, without unpredictable errors when compared to generative approaches to response creation. However, these chatbots usually have a specific and straightforward task and are not flexible enough to conduct a conversation, which is regarded as an important factor in engagement creation.

The design of the chatbot must consider all the important aspects that were referred along this section. The customisation and engagement creating features are considered of uttermost importance in the acceptability and continuous utilisation of the system, which is a key requirement in agents that focus on altering users' habits.

3.5 Summary

This chapter described the foundations of chatbot construction that are crucial to understand the articulation of the several components, namely the NLU and the Dialogue Management, which are key parts. Moreover, it mentioned and described some available tools that allow development, as well as the reasons behind the selection of Rasa as the preferred framework. Evaluation metrics were also highlighted, namely the inclusion of a user validation process. This chapter ends by describing some existing health-related chatbots and the challenges that may surge along design and development.

That being said, the next chapter introduces the scope and structure of this work, details the initial conditions and delineates the use cases that are going to be addressed by the developed conversational agent.

Chapter 4

A Chatbot for Lifana

In this chapter the addressed problem will be explained in detail including the scope of the dissertation, the available resources, as well as the selected tool and its structure.

4.1 Problem description

This section will characterise the problem in terms of tasks that should be performed, positioning in the *Lifana* ecosystem and also the available resources and the approaches taken to surpass the limitations. The focus of this work is to create a chatbot that is able to replace the graphical user interface in the *Lifana* App. Hence, the work consists on the construction of a system that is capable of encoding unstructured information from the user input (language understanding) and query the recommender system for the requested action/information. Also, the inverse flow should be contemplated: the requested information, which will be in a structured format, should be decoded and combined with natural language to output a text/spoken message (see Figure 4.1). Therefore, every possible action in the app should be mapped to a command (or several command) in the chatbot. The chatbot will have to attend to commands, similarly to what common smart assistants do (such as *Siri* or *Google Assistant*). Thus, the focus will be on the development of those command/action relations.

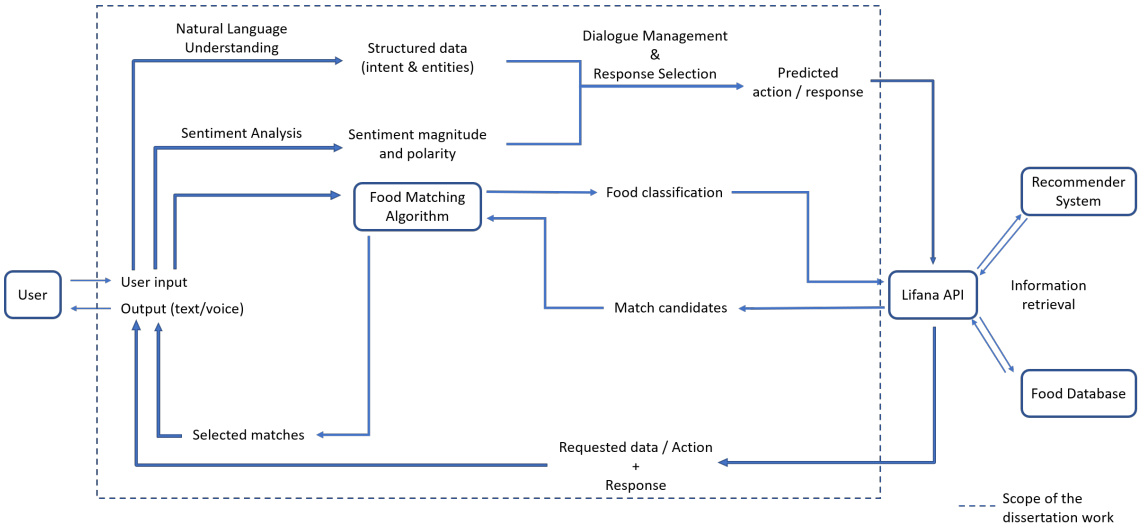


Figure 4.1: Graphical representation of the scope of the dissertation work. The agent establishes the communication between the recommender system and the user.

4.1.1 Lifana API

The *Lifana* mobile app works by retrieving and inserting information in a database through HTTP requests to a REST API. This way, the information can be accessed through various different methods such as a website, a mobile app or a programming interface. This method serves not only as the source of information but also as the means of communication between the chatbot and the recommendation engine and database. The requests' output is usually in *JSON* format, which can be parsed to extract the information needed. The API is divided in two main parts that have access to different services: the *nutritionist* part and *mealplanner* part. The *nutritionist* part deals with the nutritional information that *Fraunhofer* investigators and nutritionists collected and processed, in order to make the recommendations possible. These services are mainly used by nutritionists to complement the system and enable the following requests:

- Extended recipe information, including the possibility of query recipes by ingredient, detailed nutritional content and labelling according to dietary restrictions.
- Extended ingredient information, which includes the *LanguaL* classification among other labels and detailed nutritional content.
- Other functions related to recipe creation and nutritional needs estimation that will not be considered in the chatbot, since they are aimed at the nutritionists that work in the back-end of the application

The *mealplanner* part consists of the services addressed to the user itself. These are the services more often used by the mobile application and deal with:

- Meal planning includes the creation, visualisation and deletion of meal plans for specific time frames and/or meals. Also, the tasks related to substituting meals for alternatives and nutritional content requests for days, meals or parts of a meal belong to this group.

The nutritional content that is possible to request through these services has reduced information, presenting only the most important nutrients (carbohydrates, protein, lipids, salt, sugar, water and energy content). Furthermore, it is only possible to deal with the scaled recipes and not the basic dosage of the recipe. Recipes are scaled according to the user's nutritional needs.

- Ingredient search by name; however, it is not possible to request nutritional information.
- Personal user information management, such as the addition or alteration of demographic user data (weight, height, name, activity level, among others) or dietary restrictions (vegetarian, diabetic or vegan).
- Recipe and ingredient rating services that give a score to each ingredient or recipe. This score will be considered when creating new plans.
- Functions related to other external services such as the integration with *Fitbit* activity level estimator and with the shopping cart. These functions will not be addressed in this work.

During development, the understanding and testing of the API was made through the *Postman*[1] platform, that allowed for response visualisation and code management.

4.2 Available data

This section explores the available resources in terms of data at the beginning of the development.

4.2.1 Food related data

As stated before, Lifana recipes and ingredients database contains a lot of information regarding the ingredients and the recipes. This information will be useful to model food preferences. Recipes and ingredients are connected in a relational database by a set of primary keys in the form of identification numbers (id's). The same happens to each scaled recipe (see Section 2.2.1.1), each meal in the plan and the plan itself. It is possible to navigate through the information using these primary keys. A simplified structure of the database is presented in Figure 4.2. Most of the API requests work by providing the id of a given component either in the url or in the body of the request. Moreover, each entity has attributes. For example, each ingredient has its nutritional values and LanguaL facets that describe them according to a set of different characteristics, as mentioned in section 2.2.2. Also, each recipe has information regarding its suitability when considering dietary restrictions. The fact that each ingredient is labelled according to LanguaL facets is crucial for a proper preference modelling.

[1] https://www.postman.com/

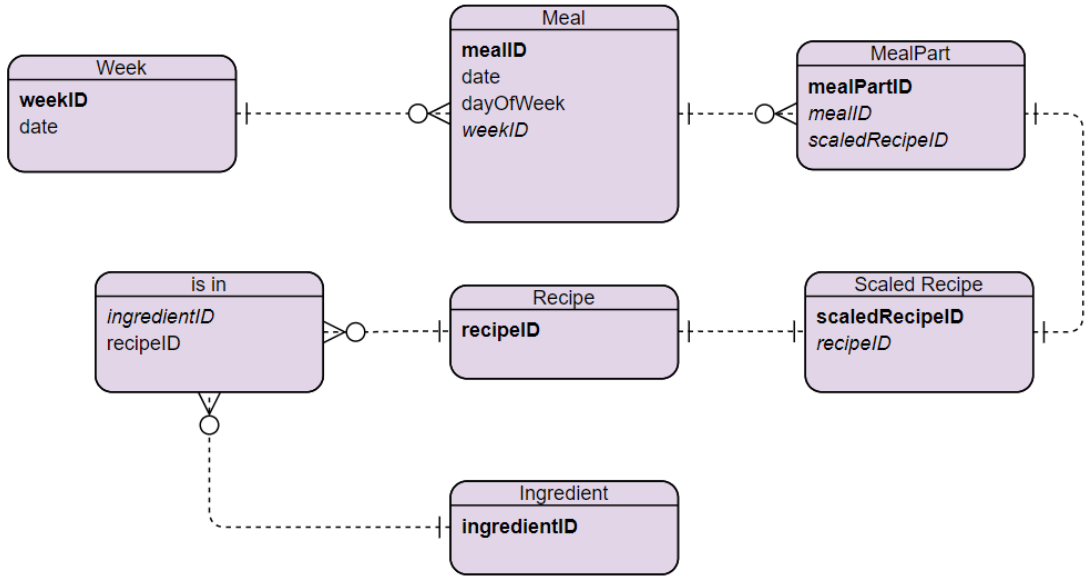


Figure 4.2: Simplified schema of the database including only the main plan elements and their relationships

It is also essential to understand the structure of the meal plans generated by the recommendation engine. Each meal plan has a week id as the main scope of the recommendation engine is to create plans for weekly time frames (the goal is to balance both the nutritional content and the type of meals along a week). Nonetheless, it is possible to have an arbitrary number of days planned for a week. The second and third levels in the structure are day and meal, respectively. The day corresponds to the day of the week and the meal corresponds to the possible meals in each day (breakfast, middle morning snack, lunch, middle afternoon snack, dinner and after dinner). Each meal is then composed by up to three meal parts, where one of them is always the main dish and the others can be the drink, the dessert, the fruit or the soup, depending on what meal is being considered. Regarding the meal parts, these correspond to the aforementioned scaled recipes which are made of ingredients. This hierarchy is very important for the development of the use cases in the agent.

4.2.2 Intial NLP data

The hardest challenge of this work was the lack of a proper dataset for the NLU tasks, namely entity recognition and intent classification. Bearing in mind that the goal is to create an intelligent agent that should understand a text input and then make a prediction on the next action to perform, data will be necessary as with every AI application. However, due to the very closed and specific domain and scope of this chatbot, there was not an available dataset for the NLU part with annotated intents or even entities, neither for the dialogue management, such as recorded conversations between a nutritionist and a patient. The major drawback of not having a proper dataset is the lack of validation in the methods that are applied to solve the different tasks. The

approach used to overcome this problem and kick-start the chatbot will be further explained in Section 5.1.1.

4.3 Rasa Framework

Taking into consideration the aforementioned resources and the fact that this framework is free and open source, the selected framework for the chatbot construction was Rasa. As referred in section 3.2.2.4, this framework uses a similar approach to Google's Dialogflow. It is based in intents and presents a way of easing the workload in terms of combining several NLP modules for different language processing tasks and allow for a focused effort on making the agent as natural and engaging as possible. In the following sections the structure and functioning of Rasa will be explained. Rasa applies a separated approach in which NLU and Dialogue management are considered independently. Therefore, it can be divided in two main parts: Rasa NLU, that deals with the Natural Language Understanding component, and Rasa Core, that handles dialogue management.

4.3.1 Rasa NLU

As the name implies, this part of the Rasa framework presents the tools needed to overcame the Natural Language Understanding tasks, namely intent identification and entity recognition.

4.3.1.1 Data format

Usually, when developing an AI agent, data is necessary to train the models, even though there are agents that do not require it (eg. reinforcement learning). In this case, Rasa NLU saves all of the text data in one Markdown[2] file, which is a plain text easy-to-read formatting syntax that can be converted into other formats like JSON. This file contains every utterance grouped by each different intent and with the entities annotated and identified. Figure 4.3 shows an example of the training data format used by Rasa.

[2]https://daringfireball.net/projects/markdown/

```
## intent:greet
- Hi
- Hey
- Hi bot

## intent:create_plan
- Plan me [breakfast](meal_division) for next [wednesday](day_meal)
- Hi! Can you please make me a plan for the [next seven days]{"entity": "period", "value": "week"}?
- Can I have a new plan for the next two [weeks](period)?
- please create me a plan for [tomorrow](period)
- Give a plan for [tomorrow](period)
- Hey, can you create a plan for the next three [weeks](period)?
- Please, create me a [weekly](period) plan for the [following week](period).

## intent:show_plan
- What is on the menu for [Monday](day_meal) and [Wednesday](day_meal) [dinner](meal_division)?
- Can I see next [Tuesday's]{"entity": "day_meal", "value": "TUESDAY"} [desserts](day_meal_part)?
- Can you show me my plan for [today's](period) [dinner](meal_division)?
- Show me what I'll have for [breakfast](meal_division) [Friday](day_meal:FRIDAY).
- Will I have any [chocolate](food) next [weekend](period)?
- What are the meals for [Monday](day_meal) [dinner](meal_division) and the [Tuesday](day_meal) [lunch](meal_division)
- What am I having for [dinner](meal_division) next [Saturday](day_meal)?
- What do eat [now](period)?
```

Figure 4.3: Illustrating example of some samples for three different intents. Note that entities are annotated using "[]", with the correspondent entity name and synonym between round brackets

It is also possible to list in this file other sources of data that complement the NLU tasks, such as Regular Expressions (regex), Synonyms and Lookup Tables. The latter are text files with a list of examples for each entity, that are used by the classification model as a feature to increase entity recognition rates.

4.3.1.2 NLP pipeline

In order to obtain the intent and entities present in each utterance, it is necessary to process the text using Natural Language Processing techniques. In Rasa, this can be defined as a pipeline of techniques that are applied to the text. The pipeline consists in several different components that generally require a specific data type and output a different one. In this case, there are four main types of data that this framework uses to ultimately classify utterances: *text*, *tokens*, *features* and *entities*. As an example, the first component of the pipeline is usually a tokeniser function that receives text and outputs tokens. Entity extractors usually receive tokens as inputs and then output the recognised entities, using also features extracted from other components. A pipeline is commonly composed of three main parts:

1. **Tokenisation** – These components often receive raw text as input and after processing it, output a list of tokens which can be words or phrases (n-grams). Rasa has default support for a set of tokenisation algorithms.

2. **Featurisation** – As the name implies, these algorithms extract token-level features that can be used in the following components, the ones that deal with the classification process. The featurisers can be divided in two groups by the type of features they produce: dense features or sparse features. A good example of a sparse feature is the BoW vector, since it has a lot of missing values (for the words of the vocabulary that are not present in the sentence). Dense features can be word embeddings or sentence-level vectors.

3. **Intent Classification / Entity Recognition** – Using the features extracted previously, these components can classify the intent of the utterance and extract its entities. After this task is done, the unstructured information in the text input has been transformed into structured data that can be used to determine the bot's next action or have the bot to extract information from a knowledge base.

This approach makes the framework extremely flexible and modular. However, the order of the components is very important for a correct operation. The components are coded using *Python* and it is possible to select them from a set of predefined ones or create custom ones using external libraries such as *NLTK* or *spaCy*. Rasa has also built-in the support for specific language model featurisers and tokenisers, such as BERT and its variations. Regarding the classification models used in the last group of components described above, Rasa presents a set of plug-and-play options that can be easily applied and tuned for each case. These options range from a BoW-based intent classification model and Conditional Random Fields (CRF) for entity recognition, to language model transformer-based classification models for both intents and entities. Further, developers at Rasa have created a transformer-based classification architecture that is able to jointly perform intent identification and entity recognition, which is worth highlighting.

- **Dual Intent and Entity Transformer**

  This architecture is based on the work of Goo et al. [170], that state that addressing these two tasks independently may increase error propagation and thus, an individual multi-task architecture should see its performance enhanced from mutual improvement between the two tasks. The Dual Intent and Entity Transformer (DIET) classifier is a modular architecture that can accept several types of word embeddings as inputs and matches current State-of-the-Art architectures in accuracy, with the major advantage of being faster both in training and inference time than large language models like BERT [171]. This last factor is crucial in conversational agents, since answer time is a very important factor to consider. In these cases it is necessary to reach a compromise between performance and timing. However, DIET can bring both since it outperforms fine-tuning BERT in benchmark NLU datasets like GLUE [172] and SuperGLUE [173]. All of the parts of this architecture can be enabled or disabled, granting the referred modular scheme. It works by receiving the list of ordered tokens extracted from previous components and encoding them using some sort of pre-trained model: either pre-trained word embeddings (eg. GloVe or Word2Vec) or pre-trained language models (e.g. BERT [90], ELMo [174] or GPT [175]). Following the methodology adopted by Devlin et al. in BERT [90], a *__CLS__* classification token is added to the end of each sequence. This token grants extra contextual information to the word vectors. The architecture then combines these vectors obtained from the previous step with sparse features extracted from each token, by concatenating them. The concatenated features pass through a feed-forward fully connected layer in order to match the size of the subsequent transformer layer. The weights of the feed-forward layer are shared among every layer. The vector that comes out of the transformer layer is then passed through a CRF tagging layer in order to predict entity labels for each token – the Named Entity Recognition part. Regarding intent classification, the transformer layer output for the *__CLS__* token is embedded into a vector space that also includes the vectors of each intent label. The model then tries to maximise the similarity between the correct label and

the token and minimise similarity between the incorrect labels and the token, by using the dot-product loss. This is the metric used to rank the intents at inference time. Moreover, this architecture also utilises a masked token approach while training. Roughly 15% of the tokens in a sentence are substituted for the token *__MASK__* which is then subject to same path of every other token. The model then embeds the both the mask token and the token that was originally unmasked and calculates a similar loss to the one used in the intent classification (dot-product loss). According to the authors [171] and based on the work of Yoshihashi et al. [176], this method of adding a masked reconstruction objective in training would serve not only as regularisation factor but also as a way to make the model learn more general features from the text, instead of only capturing discriminative features to improve classification scores. The model is trained with the aim of reducing a combined loss function that adds the three processes that were previously explained: Named Entity Recognition, Intent classification and Masked Reconstruction. As a modular approach, any of those losses may be disabled. Furthermore, all of the hyperparameters of this model can be tuned for increased performance. The architecture can be seen in Figure 4.4

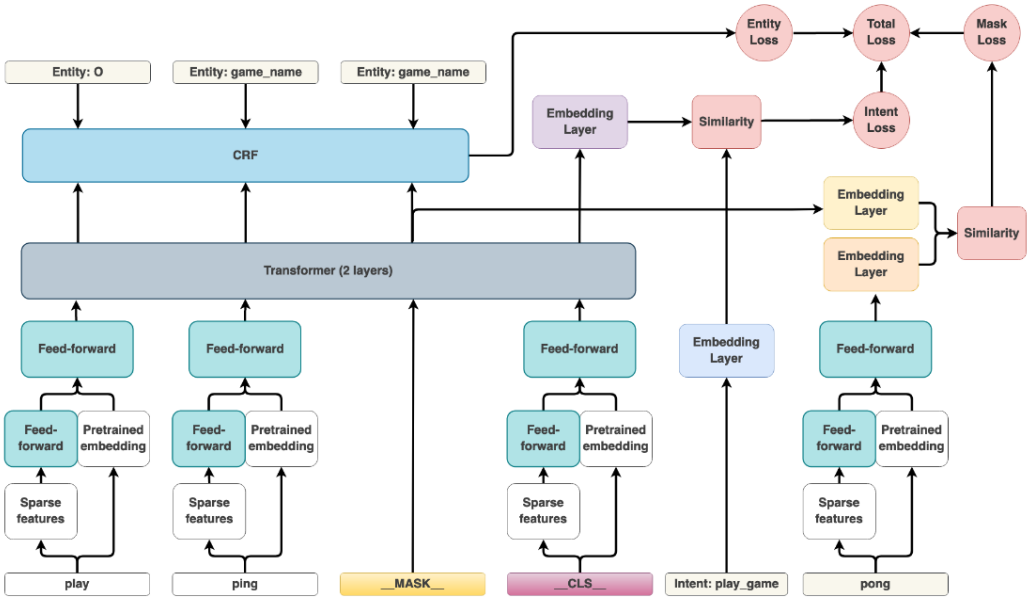


Figure 4.4: Dual Intent and Entity Transformer model architecture. The model was developed by Bunk et al. and aims at giving a state-of-the-art performance while also maintaining a reduced inference time [171]

There is also the possibility of using custom models based on the features that were extracted from the other components, such as recurrent neural networks (e.g. LSTM) that can perform well using word embeddings. Rasa framework uses a configuration file that lists what components should be used and their parameters and also their order.

4.3.2 Rasa Core

The Rasa framework has also a part dedicated to Dialogue Management with various functions, ranging from customisable actions to response selection. This part is equally in charge of the memory state of the chatbot and its connection to external data sources. It controls all the domain knowledge, allowing for the construction of a more aware and discerning agent. Similarly to the NLU part, it also requires training data. Besides this, the prediction of the next action is based on a set of policies that create a hybrid selection model, enabling both flexibility and extended control over the outcome. In the following sections all of these elements will be explored for a comprehensive understanding of the framework.

4.3.2.1 Chatbot memory

In order to keep a coherent conversation along several turns, the agent should keep record of some of the entities that are being recognised, as well as past intents. It is crucial that the bot memorises some information so that the conversation makes more sense and seems more human. For this purpose, Rasa agents have a tracker class that stores all of this information during the whole conversation. Namely, the entities can be stored in slots that are accessible by the custom actions or used as features to predict the next response. These slots can be set to auto-fill if the name of the slot matches exactly one entity type. For example, in the case of this work the agent was trained to recognise the entity *meal* which, after extraction, can be stored in the slot with the same name, seamlessly. Nevertheless, entities that were recognised during the conversation turns and were not saved in any slots, can also be used in the custom actions by navigating the tracker information, though they are not used for prediction. The slots are also important to give personal information about the user. As an illustration, the agent can be programmed to send a welcome message that can be customised using user information such as the name or gender, queried from a database or API and saved in the slots. This way the user does not have to introduce itself every time.

4.3.2.2 Data format

In the case of Dialogue Management, the aim is to predict the next step of the conversation based on the utterance that the user has introduced. Thus, it makes sense that the training data corresponds to a set of saved conversations in the form of a sequence of back-and-forth turns between the user and the agent. This format includes information about the identified intents and the predicted responses by the bot, as well as all the slots that were filled during each conversation turn. Essentially, it consists in a list of example paths that the agent can take – *stories*. As with the NLU training data, the stories are stored in a Markdown file for easy reading and writing. Figure 4.5 illustrates the data format used in this case. It is possible to notice the user intent and the triggered actions and responses (different indentation).

```
## Show Meal Part Plan Story
* start_conversation
    - action_start_conversation
    - utter_greet_name
* show_plan{"sentiment":"neu","sentiment_magnitude":0,"meal_division":"DINNER","day_meal_part":"DESSERT",
"day_meal":"MONDAY","time":"2020-06-15T00:00:00.000+01:00"}
    - slot{"day_meal":["MONDAY"]}
    - slot{"day_meal_part":["DESSERT"]}
    - slot{"meal_division":["DINNER"]}
    - slot{"sentiment":"neu"}
    - slot{"sentiment_magnitude":0}
    - slot{"time":["2020-06-15T00:00:00.000+01:00"]}
    - action_fetch_plan_date
    - slot{"period":"day"}
    - slot{"time":["2020-06-15T00:00:00.000+01:00"]}
    - slot{"meal_division":["DINNER"]}
    - action_show_plan_meal_division
    - slot{"plan":{"MONDAY":{"DINNER":{"DESSERT":["Nectarine",27147]}}}}
    - slot{"meal_ids":{"MONDAY":{"DINNER":9800}}}
    - slot{"day_meal":"MONDAY"}
    - action_day_part_plan_buttons
* inform_day_meal{"chosen_meal":"27147"}
    - slot{"chosen_meal":["27147"]}
    - utter_what_information_ingredients_nutritional
* show_nutritional_content{"sentiment":"neu","sentiment_magnitude":0}
    - slot{"sentiment":"neu"}
    - slot{"sentiment_magnitude":0}
    - action_show_nutritional_info_scaled_meal
    - utter_anything_else
```

Figure 4.5: An example of a story that shows the planned meals for a given time. It saves all the slots set at that moment as well as the intent and entities extracted by the NLU module and the bots responses. Marked with "*" are the user intents wheres the indented lines starting with "-" denote the bots actions or slots

4.3.2.3 Response prediction

Dialogue management in Rasa is based on a group of policies that control the prediction process. It is possible to include multiple different policies that articulate according to a hierarchy. There are 5 possible predefined policies to choose from in Rasa Core:

- **Keras Policy**

  As the name implies this policy includes a Recurrent Neural Network implemented in Keras that trains on the stories data and predicts the next action. Recurrent models have been the usual choice when it comes to sequential data like open [177] or close [178] domain conversational AI agents, due to their memory state capacity. These models are based on an LSTM architecture; however, it is possible to override the model for a custom one. The default model uses a softmax layer at the end to calculate the probability of each class. Also, the loss function is categorical cross-entropy and tries to maximise the accuracy value, appropriate for a multi-class classification problem. The fact that conversations may be regarded as a sequence of ordered data structures justifies the necessity of the recurrent module in the architecture. This way it takes into consideration conversation turns that happened at different timestamps. The stories from the training data are encoded into embeddings during training. This way, the model can receive a new user utterance and through a trained encoder, predict the next action.

- **Transformer Embedding Dialogue (TED) Policy**

According to the work of Vlasov et al. [179], the application of self-attention architectures enables the model to select the conversation turns that should be considered for the prediction, instead of considering them all relevant *a priori*. Therefore this type of model is more suitable for handling multi-turn conversations. As a matter of fact, RNNs should be able to predict accurately every possible scenario if a proper and large enough corpus is available for training [179]. Vlasov et al. also worked in adapting the RNN architecture to overcome an intrinsic aspect of it that is unwanted in dialogue modelling. The fact that RNNs consider the entire sequence of input data elements to generate the encoding does not help the model generalise well, unless the architecture is complex enough and has a lot of data available to learn what is actually important and what to forget. However, this is not the case for most of the conversational AI agents being developed, since they focus on very specific tasks or goals and hence there is not a lot of data available. Identically to the NLU processing pipeline for classification, Rasa developers [179] created a transformer-based architecture that is able to perform Dialogue Management, using embeddings – Transformer Embedding Dialogue Policy. This architecture builds upon the work of Vlasov et al. [180], where the classification is based on similarity of embeddings. The encoding and selection of important turns is made by the transformer that receives as input the data from previous turns, user intent and entities and the slots that were saved. This data is firstly encoded as binary vectors that indicate which intent or entities were present in the utterance and what slots are filled at that moment (one-hot encoding). During training, these vectors are then passed through the transformer and then through an embedding layer that also embeds all possible actions that the agent can execute. Then, the dot-product similarity is calculated and is the metric that drive the learning process. In addition, this structure is able to ignore short conversation turns that diverge from the goal better than Recurrent architectures. The functioning of the architecture and its structure can be found in Figure 4.6

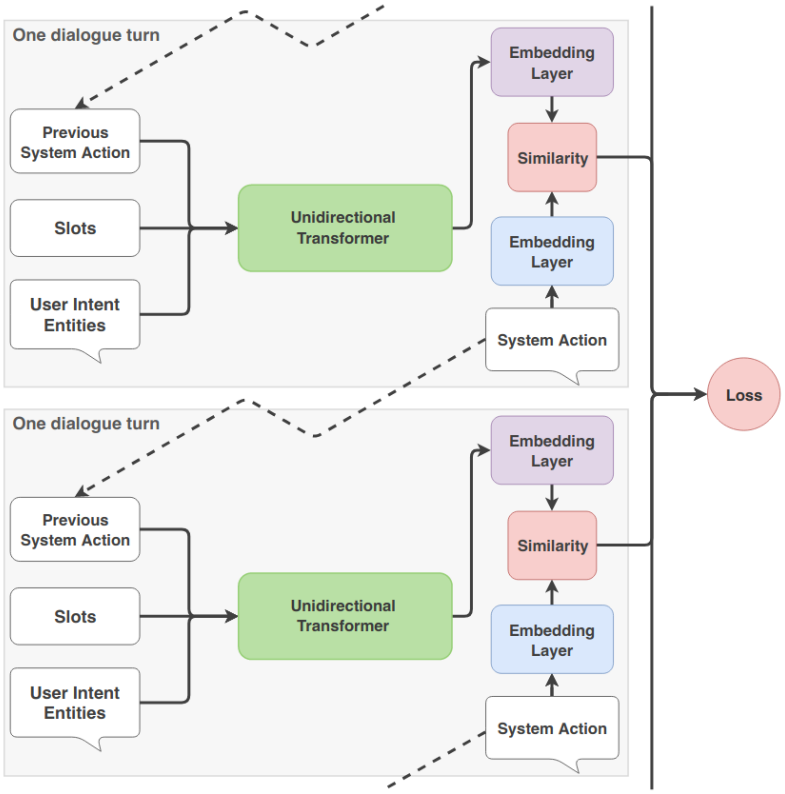


Figure 4.6: Transformer Embedding Dialogue Policy architecture and data processing along two conversation turns [179]

- **Mapping Policy**

  Additionally, the Rasa framework allows for strict action prediction with the Mapping Policy. This makes sure that a certain intent always triggers a given action or response. This can be useful for intents that are more often than not expecting a predefined answer such as a greeting or chitchat.

- **Memoisation Policy**

  This policy memorises every story in the data file and tries to reproduce its path if the current conversation matches a recorded story. This method may by programmed to forget some turns of the story and still try to predict the next step based in memorised paths. This is useful in simple cases where the user is not supposed to have a lot of freedom or the conversations can only follow a given number of paths.

- **Fallback Policy**

  Whenever the agent is not sure about the next prediction, the Fallback policy is activated and performs some sort of action that can be customised. This happens when one of these three events happen: the intent identification confidence is below the NLU threshold parameter; the highest and the second highest intent predicted have a difference in confidence that is smaller than the ambiguity threshold; or none of the policies is capable of predicting the following action with a level of confidence higher than the Core threshold. Every threshold mentioned can be changed and tuned manually in the configuration file.

Usually, the Fallback action reverts the user utterance so that it is not considered in the conversation for the prediction of the next turn.

- **Form Policy**

  In case the agent has to collect a group of entities, for example personal information for profile creation, Rasa has a tool that handles this scenario very well and in a straightforward fashion. This requires that a Form action is created in the actions file (referred in Section 4.3.2.4). This action must agree to a template that is able to loop over the required slots and sequentially ask the user to fill them. The form enables validation of the inserted information and also a way to automatically re-inquire the user for a correct information. This is useful to collect information for an API call or a database search, for example.

All of these policies can be activated in the agent, although their predictions will be ranked according to the following order, which is the default determined by Rasa, with smaller numbers having larger priority:

1. Form Policy
2. Fallback Policy
3. Memoization Policy
4. Mapping Policy
5. TED Policy and Keras Policy

Both the order and each policy can be tuned (change the hyperparameters) or overridden by a custom one. The policies should be listed in the configuration file together with the correspondent parameters.

4.3.2.4 Responses and Custom Actions

It is expected for a conversational AI agent to respond with text/speech messages. Indeed, a lot of the possible answers can be purely textual. The possible answers must be stored in a domain files that compiles all of the initial knowledge of the bot in terms of possible intents, entities, slots, actions and responses. The text responses should be listed and grouped so that the bot may choose one from the possibilities. An NLG (Natural Language Generation) model can be added to the pipeline so that the responses are generated in an external service that connects to Rasa. This option will not be explored in this work. Moreover, responses may be selected from a set of possibilities either randomly or based in a retrieval model that is similar to DIET. The last method requires extended and more detailed NLU data, in order to create smaller groups of intents inside one big intent class. For example, when dealing with different kinds of chitchat (e.g. weather questions). By separating chitchat intents in two sub-classes, the appropriate chitchat response is then selected according to the subclass of the intent. This feature will not be explored in this work, as the main focus is to handle nutrition related data. Nonetheless, as a goal-oriented agent, it must have the ability to perform actions more complex than just textual responses. These actions work exactly like the text responses when considering the dialogue

management. They are regarded as possible response to the user and can be predicted to run. In light of this, it is possible to build a collection of Python coded functions that increase the functionalities of the agent. The actions are saved in a file that the chatbot can access and execute. They are in the format of Python classes that include a *run* method that executes the code that is inside of it, in the same way as a function. However, there is one distinct feature: these actions must return a list of Rasa events. The following items explain the most important events used in this work:

- Setting slots – The output of the action may fill a slot with an entity extracted from the user input or from an external source such as a database or a request to an API. The slot being set constitutes a feature that can be used in order to predict the next step.
- Followup Action – It is also allowed to set the next action by triggering from inside the action. This is very valuable when controlling the flow of the conversation. For example, after a request to an API and according to the response there is the possibility of triggering different responses from the bot.

During the execution of the action, the chatbot may be commanded to send some sort of message to the user, although unless these messages are triggered by the Followup Action event they will not be considered for dialogue modelling. Regarding the format of the answers, they can range from pure text to images, also including buttons. The latter are a good way of presenting structured information and control the flow of the conversation, since each button has a customisable payload that can correspond directly to an intent with registered entities.

4.3.3 Extra features

Another set of features that improves the experience of using Rasa are the interactive learning mode and the proprietary deployment platform that allows public access to the chatbot.

4.3.3.1 Interactive Learning

When dealing with a low resources setting, it is important to have a straightforward and efficient way of generating preferably annotated data. Rasa interactive learning mode enables exactly this process both for NLU data and stories for dialogue modelling, although its impact is larger for stories creation. This mode runs a version of the chatbot and in each turn it asks the developer whether the identified intent, the recognised entities and the predicted next action is correct. This way, it is possible to fix some mistakes that the chatbot is making while debugging the actions and creating new data. Indeed, the conversation that the bot was having may be saved in the correspondent files. The annotated utterances are saved in the NLU data file and the conversation turns and slots are saved in the stories file.

4.3.3.2 Rasa X

As mentioned above, Rasa also has a deployment platform for the chatbot, even though it can be deployed in several platforms such as Facebook Messenger[3], Slack[4] or even the mobile platform Telegram[5]. Despite that, their platform – Rasa X – has some advantages from the development point of view. It connects to remote repositories such as Github[6] and easily updates the code that was changed locally, after being pushed to the repository. In addition, all the conversational features that can be included in the chatbot (such as image response buttons) are optimised to work in that platform. Also, there is the possibility of sending a guest user URL to a tester whose conversation with the agent is then saved and can be further annotated and saved for training, using the Rasa X interface. Another great advantage is the possibility of performing interactive learning with a graphical user interface that facilitates the process (instead of using the command line).

4.4 Use Cases Specification

Posterior to the selection of the framework and analysis of the available resources in terms of data and API, it was important to define and select the use cases of the final product and consequently the tasks that it should be capable of executing. After an extensive analysis of the functions enabled by the Lifana App and having a conversational paradigm in mind, the following tasks were selected as suitable. Some of them are not possible in the App, or require an excessive amount of effort, even though they become easier in the chatbot. The use cases are resumed in Table 4.1 and explained in the following sections. These use cases correspond to a conceptual definition, whereas the practical concretion is explored in Section 5.4.1

4.4.1 Meal Plan Actions

There is a set of possible actions regarding the meal plan handling that includes the following options:

UC.1 **Meal Plan Creation**

This is one of the main tasks in the App and should be directly mapped to the conversational agent. The user should be able to generate a meal plan for a given time period and/or meal. The time and meal are going to be extracted from the user's utterance or recorded slots. In the App, the creation of plans is either for one day at a time or for one week. Also, the creation of specific meals requires the user to deactivate all other meals in the settings of the App, proceed to the creation of the plan and reactivate them once again. This process is going to be facilitated through the use of the chatbot. It should be able to create meals for several individual days at once, such as the weekend period.

[3]https://www.messenger.com/
[4]https://slack.com/
[5]https://web.telegram.org/
[6]https://github.com/

UC.2 **Meal Plan Deletion**

The meal plan can also be deleted after creation. In this case the API only supports deletion of a complete week, so the user will only have the power to delete the plan for the whole week as well. This is not a very used function, the focus should be on other more complex and more important actions.

UC.3 **Meal Plan Visualisation**

This is probably the most complex task that the chatbot should handle. The increased level of complexity is brought by the possibility of seeing any element of the meal plan: complete days, the meals of one day, the meal parts of a given meal or even combinations of these items, such as seeing all of the desserts from two specific days in the week. The advantage of the conversation is that it enables the possibility of making requests with an extended level of specificity, when compared to the fixed and sequential navigation structure of the mobile App. Moreover, the organisation of such an output must be carefully selected so that the structure of the hierarchy is maintained and the user can easily read through all the options. Further, it must allow for navigation inside of each category if more detail is needed.

UC.4 **Alternative Meals Visualisation**

After the plan is created, the user is able to see alternative meals if that meal is not appropriate or the user wants to substitute it.

UC.5 **Meal Substitution**

This action happens always after the previously described one and, as the name implies, the user may substitute the original meal in the plan for one of the alternatives provided by the previous use case.

4.4.2 Nutritional Content Actions

Checking the nutritional information regarding the elements of the plan is also a common use case. The user may request to see the information about each one of the elements present in the plan, ranging from the day to the meal part, including the meals. Furthermore, the user may want to check the nutritional content of a given ingredient by requesting it from a meal plan or by looking for it generically. Therefore the actions are the following:

UC.6 **Check nutritional content of a day**

UC.7 **Check nutritional content of a meal**

UC.8 **Check nutritional content of a meal part**

UC.9 **Check nutritional content of an ingredient**

This one can be divided into requesting the nutritional content from an ingredient that is present in one recipe and that is being accessed at that moment or requesting the information generically about an ingredient that can have multiple values in the database (e.g. looking for cheese generically, instead of a certain type of cheese). This last task requires

an algorithm for matching the entity (ingredient) that was uttered and its values in the database. This algorithm will be addressed in Section 5.3. This action is not possible in the App; however it makes sense to implement it in the chatbot, since the information is available.

4.4.3 Food Preference Actions

This is also a very important task for the chatbot to accomplish. In the App, user preferences are modelled using a numeric rating (from 1 to 5) and control the frequency of times a given recipe or ingredient appears in the plans. However, preference modelling through conversation is not normally performed using a numeric rating. Rather, it can be done using verbs or adjectives with different levels of intensity. This challenge is going to e addressed in Section 5.2. Since it is possible to rate both ingredients and recipes, the following actions are going to be considered:

UC.10 **Express preference towards a recipe**

In this case the user can express its preference towards a meal part after checking it by referring either the name of the recipe or the corresponding meal part.

UC.11 **Express preference towards an ingredient**

This action can refer to an ingredient that is constituent of a meal part or, similarly to the nutritional content action, towards a generic ingredient. The latter implies a search in the database for the correct matches.

4.4.4 User Related Actions

There is also a part of the App that deals with the user profile. There are several details that the user must input in order to have a correct meal planning experience. The chatbot must be able to collect those personal details, present them to the user or change them later. Also, this information is important for a personalised experience where the bot refers to users by his or her name, for example. Therefore, the following actions will be considered:

UC.12 **Personal Information Collection**

As with the App, the user is guided through a series of questions in order to fill the personal information required for appropriate plan generation. This will be the welcoming process of the bot for new users

UC.13 **Personal Information Visualisation**

The user might ask to see what information the bot knows about him or her.

UC.14 **Personal Information Alteration**

User details may need to be updated in order to stay correct. For example, the weight or the dietary restrictions may change through time.

Table 4.1: Delineated Use Cases to be integrated in the chatbot

Use case group	Use cases	Description
Meal Plan Actions	UC.1 Meal Plan Creation	Generating a meal plan for a given set of parameters
	UC.2 Meal Plan Deletion	Removing the plan for a given week
	UC.3 Meal Plan Visualisation	Seeing the previously created plan
	UC.4 Alternative Meals Visualisation	Seeing a list of alternatives for a given meal
	UC.5 Meal Substitution	Selecting and substituting a meal from the plan
Nutritional Content Actions	UC.6 - UC.9 Check nutritional content	Consult the nutritional value for a day, a meal, a meal part or an ingredient
Food Preference Actions	UC.10 and UC.11 Express preference	Convey user tastes using natural language towards either a recipe or an ingredient
User Related Actions	UC.12 Personal Information Collection	Inquire user details to enable tailored meals
	UC.13 Personal Information Visualisation	Seeing the stored personal details
	UC.14 Personal Information Alteration	Updating saved personal details

4.5 Summary

This chapter presented a more in-depth description of the scope of this work including the challenges that should be overcome and the structure of the development. It defined the location of this work in the pipeline and introduces the initial conditions for development as well as the Use Cases that will be considered in the development of the conversational agent. The lack of a dataset directed some of the decisions taken along the dissertation, mainly in the validation processes.

The work will consider a total of 14 Use Cases, grouped in 4 sets, that constitute the most common scenarios of system utilisation. Some of these Use Cases will be mapped to intents that will trigger actions that perform the task that was intended by the user.

The next chapter will explore the experimental setup used to address the presented tasks (NLU, Dialogue Management, Preference Modelling and Food Matching), along with the results obtained.

Chapter 5

Chatbot Development

This chapter presents the description of the methodology taken in order to fully develop the chatbot. The work can be divided in three different stages:

- NLU component – The approach taken and the algorithms used, as well as the results obtained for intent classification and entity recognition will be analysed in this part. As stated before, these two processes are what define the NLU component of the chatbot.
- Preference Modelling – The algorithms and approaches taken for modelling preferences will be investigated along with the results.
- Dialogue Management component – The Dialogue Management component will be explored, referring also its results. More precisely, this section contains the description of the actions that map the Use Cases defined in Section 4.4, as well as the models used to predict the chatbot actions according to the conversation flow (including intents, entities and past conversation turns).

5.1 NLU approach

In this section the methodology used to address the NLU challenges will be explored along with the different results obtained from the applied procedures. This includes both intent classification and entity recognition from user input.

5.1.1 NLU Data

The first part of this section will address the NLU dataset that was collected to properly train the NLU pipeline of the agent (machine learning models). In Rasa, the processes included in the NLU component (intent classification and entity extraction) require annotated data (see Figure 4.3) consisting of several examples for each intent with the entities highlighted. As mentioned in Section 4.2.2, there was no available NLU data initially. However, the development of the chatbot was not possible without a proper dataset. This lead to the decision of collecting

a crowd-sourced dataset to bootstrap the NLU module. Hence, a dataset of the intents that represent the main use cases of the chatbot was collected. Ideally, the data collected should be annotated with both intents and entities. However, in order to ease the task of the volunteers only the food related entities were required to be annotated. The dataset was collected using a Google Form[1] that was sent to a wide community. In total, 81 volunteers completed the form, giving examples for each intent. The form was not addressed to the end-user population since intent collection is user age agnostic. The goal was to create enough examples so that a machine learning model would be able to generalise over new data. Data was collected in the English language.

The form introduced the chatbot and the recommendation system and explained the scope of the work. The form was divided in sections that corresponded to intents. The use case or intent and an example were presented at the beginning of the page and then the volunteer was asked to write one, two or three (depending on the intent) samples of the sentence that he would use to reach the described goal. From the use cases presented in Section 4.4, the ones that had a larger range of variability and the ones that were more important for the main actions of the chatbot were the ones selected for collection in the form. Here is the list of intents that were collected using the form:

- Create a plan for a given time frame and/or meal (2 examples) – The time frame and meal were up to the volunteer to decide. This way it was possible to increase variability in the combination of time and meals (maps to UC.1).
- Ask to see a plan for a given time frame and/or meal and/or meal part (3 examples) – Once again the parameters were up to the volunteer to decide. This is going to be one of the actions that the chatbot will have to perform the most, so it was asked the volunteer to give three examples (maps to UC.3).
- Ask to change the meal (2 examples) – In this section the volunteer could write a command to explicitly change an element of the plan or refer to it by the name of the recipe/present ingredients. As such, a separate text box for the food item was also present in case they refer one (maps to UC.4).
- Ask for nutritional composition (2 examples) – Once again it is possible to ask the nutritional composition (including specific nutrients such as proteins, lipids or carbohydrates) for any element of the plan or for a generically chosen ingredient. For this reason, a text box including the food item referred was included (maps to UC.6 - UC.9).
- Express preference (3 examples) – Another frequent use case will be expressing a like or a dislike towards some ingredients or recipes. In this case the volunteers were asked to give three examples of sentences that express a preference towards either an ingredient or a meal/meal part. However, in order to correctly model the preference into a numeric value, the volunteers were also required to evaluate the intensity of the preference on a scale from 0 to 5, where 0 is extremely negative and 5 is extremely positive. Additionally, in case a food item was mentioned, they were asked to repeat that food item in a separate text box so that the annotation can be made automatically (maps to UC.10 and UC.11).

[1] https://www.google.com/forms/about/

- Give information to the bot (1 example) – This would simulate the introduction of personal details or their alteration. Similarly to the previous intent, the type of information was a volunteer's decision (maps to UC.12).
- Ask to see saved personal information (1 example) – The volunteer could ask to see all of the information or specific fields such as name, birth date or dietary restrictions. The type of personal data was up to the volunteer to decide (maps to UC.13).

This way, it was possible to gather data already divided by intent and also increase the number of examples of food items (recipe names or ingredients). Gathering a diverse collection of food entities is important since it is a vast lexical group for the classifier to learn. The form (shown in Section A.1) gathered a total of 81 responses. The raw data was then analysed and curated in order to remove outliers or out of scope answers. Word misspellings were kept in order to increase model robustness against them. Also, the entities related to the meal plan (see Table 5.1) were annotated to the format accepted by Rasa (see Figure 4.3). After these processing steps, more examples were added to the ones extracted from the form. Other minor intents that do not present as much variation as the ones collected in the form were added manually to the NLU data file.

Furthermore, the analysis of the form answers allowed for increased perception of the functions that should be available in the chatbot. There were some examples that were not being considered in the initial plan of the agent. Namely, time frames such as *"now"* or *"month"*, or meal configurations such as *"next meal"*. This analysis is very important since enables the collection of the details used in the requests and that should be correctly handled by the chatbot. Incorrect or out of scope examples were added to the incorrect or chitchat intent groups. Bearing this in mind, the final dataset for the NLU training phase contains 1502 samples, divided by 19 intents and 1513 instances for 15 different entity types.

Adding up to this data, two lookup table files were included to improve entity recognition performance: a lookup table for person names extracted from *Instituto dos Registos e do Notariado* [181] with 4097 names and one with food names. The latter was extracted from the *Lifana* database and then processed to remove words that were not part of the food lexicon. The food lookup table resulted from the names of the ingredients and the names of three types of *LanguaL* facets that were considered to best describe the ingredients. The final count of terms is 1302. These three *LanguaL* descriptors will be further explored in Section 5.3.

The detailed numbers and description of the dataset can be observed in Tables 5.1 and 5.2

Table 5.1: Description of the entities used by the agent

Entity type	**Description**	**Values**	**Number of Samples**
activity_level	A field from the user profile with the several activity levels the user may have	lowest, low, medium or high	12
chosen_-alternative	Used as the payload of the buttons that show alternatives. Its purpose is to set a slot in bots memory.	id of the alternative (from database)	–
chosen_meal	Similar to chosen_alternative. It saves the meal/meal part id for visualisation or nutritional content	id of either a meal or a meal part (from database)	–
day_meal	Day of week that the user inputs, used in the actions that require a time period or plan scope (eg. plan creation, visualisation or nutritional content)	Monday... Sunday	129
meal_part	It captures the meal parts that make the plan. Used in the same scenarios as day_meal	main dish, drink, soup, fruit or dessert and synonyms	113
dietary_-restrictions	Information from the user profile used to set the restrictions in the database, enabling tailored meal planning	diabetes, vegetarian or vegan and synonyms	29
food	Any food related terms, used to express preferences or to to indicate a recipe or ingredient. A lookup table with extra examples was created due to the high variability of this entity type	Food or ingredient names (eg. cheese, meat or plums)	320
gender	Part of the user profile	male or female and synonyms	14
meal	Meal elements of the plan, used for the same actions as day_meal and meal_part	breakfast, morning snack, lunch, afternoon snack, dinner or after dinner and synonyms	236
number	Any number inputted by the user, used for personal details (height, weight), to select options and to give ratings	Any numeral either in text or numeric format (eg. sixty eight or 68)	*
nutrient	The different nutritional characteristics. It is used to check specific nutritional content fields	calories (energy), carbohydrates, lipids (fat), protein, salt, water or sugar and synonyms	131
period	It captures the time scope used for plan creation or visualisation	Values such as weekly, tomorrow, monthly or weekend. They are then mapped to one of the following: now, day, week or month	353
person_name	It saves person names and has also an increased variability and thus a lookup table with names was created	First and last person names	45
personal_-data	Different fields of the personal profile.	name, gender, height, weight, restriction, birth date or activity level and synonyms	131
time	It is used in every action that requires a time input (eg. plan creation or visualisation)	It can range from a full date to a day of week, including time intervals	*
			Total: 1513

** These entities are extracted by Duckling Extractor (Section 5.1.7.2) and thus are not annotated in the dataset*

Table 5.2: Description of the defined intents and corresponding number of samples in the dataset

Intent	Description	Number of samples
affirm	The user agrees with something or gives a positive answer	47
create_plan	The command to create a plan. May include time and meal related entities	179
see_the_ingredient	The user says he wants to see the ingredients of something	31
inform_preference	The user expresses preference towards either a plan element or a recipe/ingredient	281
show_plan	The command to show the plan or any specific element of the plan	306
show_nutritional_-content	The user asks to see the nutritional content of any element of the plan	168
give_personal_info	Used to indicate a value about any personal detail	164
deny	The user disagrees with something or gives a negative answer	21
show_alternative_-meal	The user asks to see alternatives or to substitute a given meal part	38
thank_you	The user expresses gratitude	45
see_the_meals	The user asks to see the meals of a given day or the parts of a meal	22
inform_day_meal	Used as button payload to indicate the press of it. Sets a slot with meal id	–
inform_ingredient	Used as button payload to indicate the press of it. Sets a slot with ingredient id	–
greet	Any sort of greeting	28
chitchat	Any topic that is unrelated to the theme or tasks of the chatbot	41
show_personal_info	The user asks to see the personal information that is saved in the database	90
change_personal_info	The user asks to update any field of the personal profile	11
delete_plan	The command to delete the plan. It often requires a time frame	19
bye	To say good bye to the bot	11
		Total: 1502

The following sections describe the experimental setup used to accomplish both intent classification and entity extraction With the purpose of reaching the best performance with the available data, a set of experiments using various pipelines were conducted and the results analysed. The techniques and feature sets employed are implemented in Rasa and hence some combinations are not allowed or require specific pipeline parts to work together. These techniques and features are described in the subsequent section.

5.1.2 Techniques Description

In order to create the pipeline with the best performance, several combinations using the available resources in Rasa were tried. These resources will be now explained:

- **Tokenisers**
  - *spaCy* Tokeniser – This tokeniser algorithm has increased performance over the simpler white space tokeniser that only considers the space between tokens to separate them. The algorithm implemented in the *spaCy* framework[2] uses a hierarchy of rules and heuristics to handle specific cases.
  - *BERT* Tokeniser – When using features generated by Language Models such as BERT, it may be required to employ the corresponding Tokeniser. In this case, BERT uses a tokenisation process based on WordPiece, which divide words into a set of common sub-units, improving the handling of rare or unseen words [182].
- **Featurisers**
  - *Sklearn* Count Vectors Featuriser – This algorithm works by creating a Bag-of-Words representation for the utterance based on the tokens extracted from the preceding step, using the tools provided by the *Sklearn*[3] Python library. Moreover, it can be configured to increase the feature space by adding n-grams, whose size can also be controlled. In this case, besides the token based BoW, it was also included a representation for n-grams with size from 1 to 4.
  - Lexical and Syntactic Featuriser – This algorithm extracts features based on a sliding window that moves over every token. The features extracted are described in Table 5.3 and consist mainly in word-level properties such as prefixes, suffixes or lower/upper case and syntactic features like PoS tag. The description of the features is shown in Table 5.4.
  - Regular Expression Featuriser – This component checks for matches between the tokens of the user input and the examples in the lookup tables and creates a vector of sparse features that takes the value 1 when it there is a match and the value 0 when there is no match. As na example, in the sentence "*Apple pie is my favourite dessert*", considering that both "*apple*" and "*pie*" are present in the lookup table, the resulting feature vector would be [1 1 0 0 0 0].
  - *GloVe* word vectors – The pre-trained GloVe word embeddings are used as dense features for the classifier
  - *BERT* embeddings – Similarly to GloVe embeddings, BERT embeddings are also used in some pipelines as dense features in an attempt to get better results, based on the contextual characteristic present in these embeddings
- **Classifiers**
  - Support Vector Machines (SVM) – As more traditional machine learning approach, an SVM is used as a classification model. It is a good model for simple multi-class tasks. SVM's try to separate the samples from each class using hyperplanes that maximise the margin between the several classes in higher-dimensional spaces. Rasa does not allow the combination of sparse features with SVM due to the vast feature

[2] https://spacy.io/
[3] https://scikit-learn.org/stable/

dimensional space created by sparse features. The SVM was optimised using grid search over the parameters displayed in Table A.1

- Conditional Random Fields (CRF) – Once again, as a more traditional machine learning approach, this classifier was used for entity recognition. CRF is a discriminative classification model that takes into account context (or variable neighbour state) to make predictions and that is advantageous when dealing with text data. The predictions are modelled as a graphical model with dependencies between predictions. The BILOU tagging scheme was enabled in this case
- DIET Classifier with the transformer part - This component was described in Section 4.3.1.2.
- DIET Classifier without the transformer part – This is an alteration to the previous classifier where the transformer layers are disabled. This means that only feed-forward fully connected layers are available for training and classification.

Table 5.3: Sparse features extracted for entity recognition using CRF. The description is in Table 5.4

Previous token	Interest token	Next token
low, title, upper	low, title, upper, BOS, EOS, preffix5, preffix2, suffix5, suffix3, suffix2, pos	low, title, upper

Table 5.4: Description of the sparse features used for entity recognition

Feature	Description
low	Checks if the token is lower case.
upper	Checks if the token is upper case.
title	Checks if the token starts with an uppercase character and all remaining characters are lowercased
digit	Checks if the token contains just digits.
prefix5	Take the first five characters of the token.
prefix2	Take the first two characters of the token.
suffix5	Take the last five characters of the token.
suffix3	Take the last three characters of the token.
suffix2	Take the last two characters of the token.
suffix1	Take the last character of the token.
pos	Take the Part-of-Speech tag of the token

The next sections describe the approaches employed that use combinations of the above-mentioned techniques and features.

5.1.3 Intent Classification

The first task that must be addressed in the development of the NLU part of the chatbot is creating a model that is able to classify user utterances into one of the intents referred in Table 5.2. This problem is a multi-class classification task with 19 classes. Class distribution is not even

because most of the examples for the intents were created through the form described in Section 5.1.1, which addressed only the most important intents, as well as the ones that may have increased variability. The approaches are resumed in Table 5.5. Note that the employed DIET classifier does not have the CRF tagger part, responsible for entity extraction, since the goal is to perform intent classification

5.1.3.1 Approaches

The described approaches are divided by the feature sets they employ: BoW (sparse features), pre-trained word embeddings (dense features), Language Model (dense features) and a mixed approach (sparse and dense features).

In an effort to create a baseline score, the simplest model based on a Bag-of-Words algorithm was implemented for intent classification. This pipeline correspond to a Bag-of-Words with DIET Classifier (transformer disabled) – I.A1. It is expected to perform the worst from all tried pipelines.

The second pipeline used (I.A2) was very similar to the one presented above, however, the transformer architecture in the DIET Classifier was enabled, which is expected to increase performance, since the transformer part is supposed to select the important features based on performance – Bag-of-Words with DIET Classifier (transformer enabled).

After creating a baseline using sparse features, with the aim of improving performance, another approach was considered using more recent and complex resources, namely dense features from word embeddings. The following pipelines use GloVe pre-trained vectors from the spaCy model. Besides this, different classification models were tried: the DIET classifier with and without the transformer, and an SVM. These pipelines correspond to id's I.B1 – GloVe vectors with DIET Classifier (transformer disabled), I.B2 – GloVe vectors with DIET Classifier (transformer enabled) and I.B3 – GloVe vectors with SVM.

Bearing in mind the increased performance of language models in NLP tasks, it makes sense to implement a pipeline that takes advantage of them. For that reason, a set of pipelines using a pre-trained BERT model from Hugging Face and several different classifiers were tested, namely pipelines I.C1 – BERT embeddings with DIET Classifier (transformer disabled), I.C2 – BERT embeddings with DIET Classifier (transformer enabled) and I.C3 – BERT embeddings with SVM

Besides these simpler approaches that rely only in one type of features, it makes sense to combine them in a more complex pipeline. Thus, a group of variations of this approach was implemented following the same logic of the previous ones. BERT Tokeniser is used in all procedures since it is a requirement to extract BERT word embeddings. These pipelines correspond to id's I.D1 – BERT embeddings plus BoW with DIET Classifier (transformer disabled), I.D2 – BERT embeddings plus BoW with DIET Classifier (transformer enabled) and I.D3 – BERT embeddings plus BoW with SVM.

Table 5.5: Pipelines tried for intent classification

Approach	Id	Tokeniser	Featurisers	Classifier
Baseline – BoW	I.A1	spaCy	Sklearn Count Vectors	DIET (transformer disabled)
	I.A2	spaCy	Sklearn Count Vectors	DIET (transformer enabled)
Pre-trained Word Embeddings	I.B1	spaCy	GloVe word vectors	DIET (transformer disabled)
	I.B2	spaCy	GloVe word vectors	DIET (transformer enabled)
	I.B3	spaCy	GloVe word vectors	SVM
Language Model	I.C1	BERT	BERT embeddings	DIET (transformer disabled)
	I.C2	BERT	BERT embeddings	DIET (transformer enabled)
	I.C3	BERT	BERT embeddings	SVM
Mixed	I.D1	BERT	Sklearn Count Vectors + BERT embeddings	DIET (transformer disabled)
	I.D2	BERT	Sklearn Count Vectors + BERT embeddings	DIET (transformer enabled)
	I.D3	BERT	Sklearn Count Vectors + BERT embeddings	SVM

5.1.3.2 Experimental Evaluation

Each pipeline from Table 5.5 was trained 3 times using 2 different train/test ratios (80/20 and 70/30). The goal of trying different train/test ratios was to compare the robustness of the model when reducing training data. Table 5.6 resumes the mean results (with standard deviation) obtained for each experiment concerning intent identification. It presents both macro and weighted averages for precision, recall and f1 score, as well as accuracy. The results also consider the different test ratios.

At a first glance, it is possible to observe that the model with the highest overall accuracy is the one corresponding to configuration I.D3, which combines both BoW features with contextual embeddings and uses SVM as classifier. As a matter of fact, approaches using SVM had higher accuracy values among all combinations. The reason behind this may be fact that the dataset was small and hence, deep learning models (DIET with and without transformer) tend to overfit more easily, which reflects on the results. Regarding both deep learning models, an interesting pattern arises: the transformer based architecture provides better overall results that the one with the transformer disabled in the configurations that use BoW features, whereas the opposite happens with only dense features. This may be due to the way the transformer architecture was developed. The transformer module may be using its attention mechanism to properly select features from the wide range of sparse features provided. Moreover, the results obtained by the transformer model improve severely when using sparse features compared to dense features only.

The set of features that provided the worst overall results are the pre-trained word embeddings from GloVe, even though the deep learning model without the transformer performed actually better with GloVe embeddings than when using only sparse features. These results may also be explained by the existence of typographical errors in the training sentences, which hinder the attribution of a correct embedding to a given token. This does not happen with the

BERT language model vectors due to the fact that the tokeniser is based on WordPieces, which decreases the number of tokens without a correct vector. All models perform the best when using both sparse and dense features combined, namely the DIET with transformer enabled which sees a boost in performance, reaching values that are close to the ones obtained by the SVM. Another interesting factor that is worth mentioning is the slim difference between the overall results of the SVM with both sparse and dense features (I.D3) and the SVM with BERT's word embeddings only (I.C3), which suggests that the sparse features are only giving a slight improvement. Howbeit, this does not hold when comparing I.B3 (GloVe features) to either I.C3 or I.D3, which shows that contextual embeddings provide better insights for intent classification than pre-trained static embeddings.

There is a noticeable gap between the metrics calculated using a macro average and a weighted average. This is due to the class imbalance nature of the dataset. Actually, this shows that the models are giving increased importance to the classes with more samples, leading to lower scores in the ones with less samples. When looking at the averages for F1 scores, it is possible to observe that both I.D3 and I.D2 are the models that show a smaller gap between the macro and the weighted values. This may indicate that an increased amount of features may help the model differentiate between low occurrence samples.

Furthermore, recall values are lower than precision scores when considering the macro average, although the opposite happens when considering the weighted average. This shows that recall values for classes with few examples is low, which implies that the model does not have very much confidence when dealing with these classes. It only classifies them when it has a high level of certainty, leading to higher precision in those classes.

Both I.D2 and I.D3 are good candidates for the intent identification pipeline of the agent.

Table 5.6: Results for intent identification using the different configurations described above

Model	Test percentage	Macro Average			Weighted Average			Accuracy
		Precision	Recall	F1	Precision	Recall	F1	
I.A1	20	0.684 ± 0.047	0.638 ± 0.054	0.634 ± 0.052	0.825 ± 0.015	0.805 ± 0.013	0.802 ± 0.015	0.805 ± 0.059
	30	0.674 ± 0.043	0.617 ± 0.075	0.618 ± 0.068	0.786 ± 0.044	0.769 ± 0.043	0.760 ± 0.044	0.769 ± 0.048
I.A2	20	0.738 ± 0.040	0.668 ± 0.027	0.677 ± 0.015	0.843 ± 0.005	0.835 ± 0.007	0.828 ± 0.005	0.835 ± 0.041
	30	0.703 ± 0.041	0.605 ± 0.063	0.628 ± 0.062	0.803 ± 0.035	0.801 ± 0.035	0.790 ± 0.035	0.801 ± 0.033
I.B1	20	0.710 ± 0.026	0.620 ± 0.025	0.642 ± 0.009	0.837 ± 0.011	0.847 ± 0.012	0.833 ± 0.011	0.847 ± 0.034
	30	0.541 ± 0.100	0.494 ± 0.087	0.492 ± 0.087	0.760 ± 0.027	0.791 ± 0.041	0.762 ± 0.027	0.791 ± 0.110
I.B2	20	0.343 ± 0.036	0.236 ± 0.037	0.234 ± 0.039	0.514 ± 0.063	0.472 ± 0.080	0.399 ± 0.063	0.472 ± 0.050
	30	0.290 ± 0.090	0.210 ± 0.023	0.201 ± 0.038	0.467 ± 0.017	0.445 ± 0.032	0.367 ± 0.017	0.445 ± 0.107
I.B3	20	0.712 ± 0.044	0.622 ± 0.026	0.643 ± 0.027	0.828 ± 0.011	0.834 ± 0.005	0.821 ± 0.011	0.834 ± 0.063
	30	0.697 ± 0.015	0.614 ± 0.031	0.636 ± 0.019	0.818 ± 0.020	0.825 ± 0.019	0.813 ± 0.020	0.825 ± 0.021
I.C1	20	0.529 ± 0.015	0.481 ± 0.011	0.478 ± 0.009	0.761 ± 0.005	0.805 ± 0.005	0.770 ± 0.005	0.805 ± 0.019
	30	0.407 ± 0.050	0.444 ± 0.048	0.414 ± 0.045	0.712 ± 0.014	0.781 ± 0.018	0.740 ± 0.014	0.781 ± 0.058
I.C2	20	0.559 ± 0.058	0.468 ± 0.024	0.476 ± 0.032	0.752 ± 0.019	0.793 ± 0.023	0.754 ± 0.019	0.793 ± 0.050
	30	0.516 ± 0.034	0.447 ± 0.020	0.448 ± 0.031	0.726 ± 0.013	0.768 ± 0.017	0.730 ± 0.013	0.768 ± 0.037
I.C3	20	0.779 ± 0.013	0.672 ± 0.036	0.701 ± 0.021	0.863 ± 0.015	0.865 ± 0.015	0.856 ± 0.015	0.865 ± 0.018
	30	0.727 ± 0.055	0.631 ± 0.053	0.656 ± 0.053	0.839 ± 0.013	0.848 ± 0.016	0.835 ± 0.013	0.848 ± 0.078
I.D1	20	0.720 ± 0.033	0.672 ± 0.018	0.674 ± 0.015	0.861 ± 0.020	0.855 ± 0.018	0.848 ± 0.020	0.855 ± 0.012
	30	0.686 ± 0.083	0.640 ± 0.037	0.640 ± 0.056	0.834 ± 0.013	0.830 ± 0.015	0.821 ± 0.013	0.830 ± 0.114
I.D2	20	0.770 ± 0.028	0.688 ± 0.050	0.703 ± 0.043	0.867 ± 0.013	0.862 ± 0.016	0.853 ± 0.013	0.862 ± 0.031
	30	0.687 ± 0.060	0.615 ± 0.046	0.629 ± 0.052	0.815 ± 0.008	0.832 ± 0.012	0.814 ± 0.008	0.832 ± 0.078
I.D3	20	0.782 ± 0.006	0.675 ± 0.043	0.704 ± 0.030	0.865 ± 0.019	0.866 ± 0.021	0.857 ± 0.019	0.866 ± 0.008
	30	0.792 ± 0.076	0.642 ± 0.068	0.664 ± 0.070	0.842 ± 0.016	0.852 ± 0.021	0.839 ± 0.016	0.852 ± 0.092

5.1.4 Entity Recognition

The second task that the chatbot must be able to complete in terms of NLU is the recognition and extraction of entities from user utterance. This is a sequence labelling problem with 15 classes. The dataset used in the training session is described in Table 5.1.

A comparable approach to the one taken in intent classification was adopted for entity recognition. A set of different combinations of feature extractors and classification models were tested in order to understand which one provided the best results. The several pipelines are resumed in Table 5.7.

5.1.4.1 Approaches

Once again, as a baseline, the simpler approach using sparse vectors was implemented. Moreover, two classification models were tested for this set of features: the CRF tagger and the DIET with the entity extraction mode activated and the intent mode disabled. In this case, the transformer part of the DIET classifier is enabled, since otherwise it would be very similar to using only the CRF. The pipelines E.A1 – Sparse features with CRF tagger and E.A2 – Sparse features with DIET Classifier (transformer enabled) from Table 5.7 were tried.

Using dense features, such as word vectors, may increase performance in Named Entity Recognition. Due to this fact, two methods that take advantage of pre-trained GloVe embeddings (equal to the ones used in intent classification) were implemented. Concerning the classification models, the configurations were kept from the previous approach. The pipelines E.B1 – GloVe vectors with CRF tagger and E.B2 – GloVe vectors with DIET Classifier (transformer enabled) were tried in this case.

Pre-trained language model embeddings can equally be used for entity extraction. For this reason, two pipelines that apply this technique were implemented: E.C1 – BERT embeddings with CRF tagger and E.C2 – BERT embeddings with DIET Classifier (transformer enabled).

Similarly to the intent classification, a methodology that joins both feature types were tested. In this case, the sparse features are similar to the ones used in the first approach, while the dense features come from the BERT model. The following variations were implemented: E.D1 – Sparse features and BERT embeddings with CRF tagger and E.D2 – Sparse features and BERT embeddings with DIET Classifier (transformer enabled)

Table 5.7: Pipelines tried for entity recognition

Approach	Id	Tokeniser	Featurisers	Classifier
Baseline	E.A1	spaCy	Lexical and Syntactic Featuriser	CRF
	E.A2	spaCy	Lexical and Syntactic Featuriser	DIET (transformer enabled)
Pre-trained Word Embeddings	E.B1	spaCy	GloVe word vectors	CRF
	E.B2	spaCy	GloVe word vectors	DIET (transformer enabled)
Language Model	E.C1	BERT	BERT embeddings	CRF
	E.C2	BERT	BERT embeddings	DIET (transformer enabled)
Mixed	E.D1	BERT	Lexical and Syntactic Featuriser + BERT embeddings	CRF
	E.D2	BERT	Lexical and Syntactic Featuriser + BERT embeddings	DIET (transformer enabled)

5.1.4.2 Experimental Evaluation

The experimental conditions were similar to intent classification. The pipelines were trained and tested three times with two different train/test ratios (80/20 and 70/30). The results displayed in Table 5.8 show the average performance (and standard deviation) obtained by the several combinations of features and classifiers regarding entity recognition. An evident pattern is the superior overall results of the configurations using the transformer architecture. This architecture considers the whole sentence where the entity is present and selects the best features from the set of available features, whereas the CRF extractor only considers the token before and after the interest token. The attention module may play a crucial role in selecting the features and capturing context.

Interestingly, the best results were obtained using the set of dense features provided by GloVe vectors (E.B2), which provided the worst results for intent identification (as reported in Table 5.6). In a closed domain where the entity types are very different, which is the case of this work, non-contextual embeddings may help the classification model to better learn the patterns and further increase performance. In these cases, context is not crucial for entity extraction. However, both the results obtained using the sparse features only (E.A2) and the combination of sparse and contextual word embeddings (E.D2) provided scores that are very close to the best values. This may suggest that the sparse features are in fact very effective for entity recognition, when comparing to approach E.C2 which uses only the contextual embeddings.

Similarly to what happened in intent identification, macro scores are lower than weighted scores, which is explained by the fact that the model actually fails more in classes with less examples. Another intriguing aspect is the F1 score macro average for both configuration E.B2 and E.D2, which is higher for the test that used less training. This showed that the model performs better in the classes with less examples when trained in less data, suggesting some level of overfitting.

Table 5.8: Results for entity recognition using the different configurations described above

Model	Test percentage	Macro Average			Weighted Average		
		Precision	Recall	F1	Precision	Recall	F1
E.A1	20	0.748 0.037	0.614 0.013	0.657 0.022	0.887 0.029	0.801 0.018	0.833 0.018
	30	0.696 0.025	0.579 0.046	0.622 0.038	0.875 0.035	0.791 0.024	0.825 0.024
E.A2	20	0.775 0.065	0.767 0.055	0.767 0.062	0.878 0.034	0.879 0.029	0.876 0.029
	30	0.794 0.039	0.734 0.072	0.756 0.062	0.878 0.044	0.847 0.046	0.859 0.046
E.B1	20	0.781 0.032	0.761 0.031	0.759 0.033	0.898 0.023	0.841 0.024	0.865 0.024
	30	0.770 0.025	0.747 0.052	0.745 0.041	0.888 0.027	0.838 0.021	0.857 0.021
E.B2	20	0.843 0.032	0.782 0.048	0.801 0.048	0.905 0.059	0.877 0.056	0.888 0.056
	30	0.877 0.022	0.782 0.068	0.806 0.050	0.915 0.051	0.866 0.043	0.884 0.043
E.C1	20	0.844 0.031	0.699 0.044	0.739 0.060	0.927 0.025	0.820 0.037	0.856 0.037
	30	0.837 0.047	0.638 0.019	0.688 0.035	0.922 0.019	0.786 0.031	0.828 0.031
E.C2	20	0.867 0.062	0.770 0.111	0.793 0.091	0.893 0.045	0.848 0.035	0.860 0.035
	30	0.839 0.100	0.725 0.077	0.744 0.113	0.905 0.018	0.823 0.032	0.843 0.032
E.D1	20	0.785 0.072	0.648 0.033	0.692 0.041	0.910 0.022	0.804 0.029	0.844 0.029
	30	0.804 0.025	0.576 0.006	0.639 0.006	0.939 0.031	0.791 0.030	0.841 0.030
E.D2	20	0.794 0.059	0.736 0.036	0.746 0.048	0.894 0.029	0.873 0.030	0.875 0.030
	30	0.880 0.046	0.768 0.040	0.790 0.058	0.880 0.007	0.861 0.016	0.859 0.016

5.1.5 Joint Intent Classification and Entity Extraction

As stated in Section 4.3.1.2, the DIET classification model is capable of performing combined intent classification and entity extraction, using the sum of both loss functions to optimise the model. The hypothesis that is going to be tested with these additional configurations is that the performance of the two processes is improved if trained together.

5.1.5.1 Approaches

In light of what was stated above, the featurising pipelines that achieved the best results overall were altered to test the combined classification methodology. The set of experimented pipelines is shown in Table 5.9. The pipelines tried were IE.A1 – Sparse features with DIET Classifier (transformer enabled), IE.A2 – GloVe vectors with DIET Classifier (transformer enabled), IE.A3 – BERT embeddings with DIET Classifier (transformer enabled) and IE.A4 – Sparse features and BERT embeddings with DIET Classifier (transformer enabled)

Table 5.9: Pipelines tried for joint intent classification and entity recognition

Approach	Id	Tokeniser	Featurisers	Classifier
Baseline	IE.A1	spaCy	Sklearn Count Vectors + Lexical and Syntactic Featuriser	DIET (transformer enabled)
Pre-trained Word Embeddings	IE.A2	spaCy	GloVe word vectors	DIET (transformer enabled)
Language Model	IE.A3	BERT	BERT embeddings	DIET (transformer enabled)
Mixed	IE.A4	BERT	Sklearn Count Vectors + Lexical and Syntactic Featuriser + BERT embeddings	DIET (transformer enabled)

5.1.5.2 Experimental Evaluation

In order to maintain the patter, the same training and testing conditions were applied. The model was trained and tested three times using two different train/test ratios (80/20 and 70/30). By taking a look at the results in Table 5.10 (average scores with standard deviation), it is possible to see that the multitask setting helped improve the performance of the entity recognition process in all configurations. Regarding intent classification, a similar thing happened except for the configuration IE.A4, which uses the combination of both sparse and dense features. In this case, the decrease may be caused by the extra sparse feature set for entity recognition that the model has to deal with. This may increase the size of feature space and hence favour overfitting, therefore decreasing the baseline (IE.A1) performance. Once again, GloVe vectors have shown to be the best set of features for entity recognition. This may be due to the unchanging values that the vectors assume, which is favourable for a closed domain such as the scope of this work. This means that there is not enough variation in word meaning for the contextual embeddings to make a difference in entity recognition. This improved performance in one task (entity extraction) seemed to leverage the scores obtained in intent classification. This is mainly visible in configurations IE.A1 and IE.A2, while configuration IE.A3 showed a slight increase. Comparing the entity extraction results for configuration IE.A1 and IE.A3 reinforces the premise that in this closed domain, these less changing features provide better results, since there is only a small improvement from the sparse features to contextual embeddings.

However, the contextual embeddings prove to be better for intent classification (approach IE.A3), where more variation may be present and context plays a more important role. Nevertheless, by looking at these results it becomes clear that the regularisation added by combining the two tasks improves performance on both tasks.

Table 5.10: Results obtained for joint entity extraction and intent classification using DIET classifier

	Model	Test Percentage	Macro Average			Weighted Average			Accuracy
			Precision	Recall	F1	Precision	Recall	F1	
Intent	IE.A1	20	0.765 0.061	0.707 0.043	0.724 0.046	0.857 0.024	0.861 0.025	0.854 0.025	0.861 0.024
		30	0.738 0.068	0.716 0.094	0.714 0.090	0.840 0.035	0.840 0.039	0.834 0.039	0.840 0.035
	IE.A2	20	0.729 0.019	0.732 0.034	0.724 0.025	0.862 0.009	0.868 0.008	0.862 0.008	0.868 0.009
		30	0.685 0.035	0.690 0.049	0.679 0.042	0.851 0.011	0.861 0.011	0.853 0.011	0.861 0.011
	IE.A3	20	0.787 0.048	0.756 0.049	0.755 0.041	0.890 0.020	0.886 0.021	0.883 0.021	0.886 0.020
		30	0.749 0.064	0.730 0.038	0.730 0.046	0.874 0.033	0.882 0.029	0.875 0.029	0.882 0.033
	IE.A4	20	0.683 0.075	0.623 0.102	0.632 0.092	0.824 0.029	0.820 0.036	0.811 0.036	0.820 0.029
		30	0.717 0.015	0.679 0.012	0.679 0.023	0.832 0.027	0.834 0.021	0.825 0.021	0.834 0.027
Entity	IE.A1	20	0.890 0.014	0.874 0.044	0.875 0.024	0.895 0.022	0.874 0.004	0.881 0.004	–
		30	0.854 0.050	0.801 0.079	0.805 0.025	0.877 0.047	0.857 0.012	0.861 0.012	–
	IE.A2	20	0.878 0.025	0.870 0.006	0.867 0.021	0.922 0.026	0.888 0.005	0.902 0.005	–
		30	0.834 0.037	0.810 0.008	0.806 0.034	0.922 0.002	0.874 0.017	0.894 0.017	–
	IE.A3	20	0.861 0.013	0.715 0.064	0.767 0.039	0.901 0.038	0.828 0.026	0.855 0.026	–
		30	0.884 0.039	0.716 0.040	0.771 0.010	0.911 0.037	0.820 0.022	0.855 0.022	–
	IE.A4	20	0.879 0.039	0.800 0.024	0.819 0.038	0.922 0.000	0.874 0.011	0.892 0.011	–
		30	0.869 0.056	0.754 0.049	0.795 0.013	0.915 0.020	0.859 0.004	0.882 0.004	–

5.1.6 Regex Features

In Section 4.3.1.1 it was referred that entity extraction could be improved using Regular Expression features, namely exploiting Lookup Tables. In fact, as mentioned in Section 5.1.1, two Lookup Tables were created to further enhance the performance of the chatbot when handling

large lexicon entities: person names and food names. With a view to assess the variation in performance, two pipelines using these features were also tried. These pipelines use the Regular Expression featuriser described in Section 5.1.2.

5.1.6.1 Approaches

In this case, only two pipelines were tried, which correspond to the best scoring ones with the addition of the Regex features. This means that the pipelines are similar to IE.A2 and IE.A4. They are: IE-RG.A2 – Sparse features and BERT embeddings with DIET Classifier (transformer enabled) + Lookup Tables and IE-RG.A2 – Sparse features and BERT embeddings with DIET Classifier (transformer enabled) + Lookup Tables. The pipelines are resumed in Table 5.11

Table 5.11: Pipelines tried for joint intent classification and entity recognition with Lookup table features

Approach	Id	Tokeniser	Featurisers	Classifier
Pre-trained Word Embeddings	IE-RG.A1	spaCy	GloVe word vectors + Regex Featuriser	DIET (transformer enabled)
Mixed	IE-RG.A2	BERT	Sklearn Count Vectors + Lexical and Syntactic Featuriser + BERT embeddings + Regex Featuriser	DIET (transformer enabled)

5.1.6.2 Experimental Evaluation

The training and testing conditions for these pipelines is equal to the previous (3 times with 70/30 and 80/20 train/test ratios). Table 5.12 resumes the results obtained for the aforementioned configurations. As expected, there is a slight increase in the F1 scores for entity recognition in both configurations, when compared to the same pipelines without the features from the lookup tables (IE.A2 and IE.A3 in Table 5.10). This increase is even more evident by analysing the individual scores of the entities for which a lookup table was created. In fact, Table 5.13 shows that F1 values for both entities increased in the two models. Moreover, it is possible to notice that the main cause of this improvement is an increased value of recall. This means that, by using lookup tables, the model is able to improve recognition regarding the positive samples, leaving less positive samples out. The gap is larger for the *person_name* entity, probably due to the fact that the training data had less samples for names and the lookup table is more extensive when comparing to the *food* entities.

The addition of the lookup tables does in fact improve the results for entity recognition. However, it also helped both models to increase intent classification accuracy. This reinforces that the dual task methodology presents benefits for the overall performance of the model.

Table 5.12: Entity extraction and intent classification results for the configurations using lookup tables as a way to improve performance

	Model	Percentage excluded	Macro Average			Weighted Average			Accuracy
			Precision	Recall	F1	Precision	Recall	F1	
Intent	IE-RG.A1	20	0.796 0.002	0.754 0.039	0.763 0.032	0.885 0.013	0.883 0.013	0.880 0.013	0.883 0.012
		30	0.795 0.010	0.754 0.039	0.761 0.002	0.880 0.002	0.880 0.003	0.875 0.003	0.880 0.005
	IE-RG.A2	20	0.776 0.061	0.714 0.029	0.728 0.044	0.868 0.018	0.866 0.020	0.860 0.020	0.866 0.011
		30	0.741 0.027	0.682 0.067	0.696 0.054	0.850 0.018	0.848 0.019	0.843 0.019	0.848 0.022
Entity	IE-RG.A1	20	0.903 0.000	0.858 0.037	0.870 0.024	0.906 0.006	0.891 0.008	0.896 0.008	–
		30	0.836 0.102	0.796 0.054	0.807 0.079	0.892 0.016	0.877 0.022	0.882 0.022	–
	IE-RG.A2	20	0.877 0.054	0.830 0.011	0.839 0.027	0.905 0.051	0.898 0.036	0.896 0.036	–
		30	0.828 0.035	0.748 0.041	0.769 0.012	0.900 0.023	0.868 0.010	0.876 0.010	–

Table 5.13: Individual performance scores for the entities affected by the addition of a lookup table (*food* and *person_name*)

Model	Entity	Precision	Recall	F1
IE.A2	food	0.927 0.048	0.860 0.033	0.891 0.006
	person_name	0.754 0.041	0.912 0.030	0.824 0.010
IE.A4	food	0.901 0.020	0.861 0.002	0.880 0.011
	person_name	0.736 0.233	0.681 0.027	0.701 0.118
IE-RG.A1	food	0.919 0.044	0.904 0.013	0.911 0.015
	person_name	0.826 0.104	0.909 0.193	0.865 0.146
IE-RG.A2	food	0.910 0.006	0.902 0.085	0.905 0.038
	person_name	0.801 0.215	1.000 0.000	0.883 0.127

5.1.7 Extra Components

In order to increase the performance and improve experience of the chatbot, as well as facilitating some tasks that will be further explained, some additional NLU components were added to the pipeline. These components do not affect the classification scores. Besides the Tonkeniser, Featurisers and Classification model, the components explored in the following sections were added to the final approach.

5.1.7.1 Spell Checker

A good experience with a conversational agent requires appropriate error handling. Namely, when considering text input, there is the possibility of misspelled words being sent to the chatbot. On the one hand, the model should be robust enough to deal with unknown words, given not only to the fact that some of the lexical features would remain equal but also to the model's ability of inferring the sense from the surrounding words. On the other hand, depending on the misspell, entity extraction could be largely affected, leading to a failure in addressing user requests. For that reason, it makes perfect sense to include in the pipeline a pre-processing step that tries to deal with typographical errors before tokenisation even occurs. Bearing this in mind, the spell checking module was implemented on top of the pipeline using the *SymSpell* Python library[4]. It works by comparing every word of the sentence with a vocabulary model through Levenshtein Distance. This distance metric calculates the difference between words on a character base. The vocabulary is based on the *Google Book N-gram Viewer*[5] and *SCOWL – Spell*

[4]https://github.com/wolfgarbe/SymSpell
[5]http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

Checker Oriented Word Lists[6]. The first is a collection of words and respective frequency among a large corpus of books, whereas the latter is a dictionary of English words. The author of the library created the final dictionary by intersecting the two lists. Given a misspelled word W, it calculates the closest options according to Levenshtein distance. It computes every possibility for an edit distance of up to n, where n is the maximum Levenshtein distance from the word to any candidate. After that step it applies a filter to select only words from the dictionary and then calculates the conditional probability of W corresponding to any of the candidates, provided the edit distance and the word frequency. This algorithm is very fast, reaching a speed of 5000 words per second, meaning that it can be used at inference time, during conversation. Moreover, it can deal with inserted or deleted spaces between tokens, by analysing bi-grams compounds, which is a great advantage over other common spellchecking algorithms, like Norvig's *pyspellchecker*[7] or Atkinson's *GNU Aspell*[8].

5.1.7.2 *Duckling* Entity Extractor

Duckling[9] is an entity extractor developed by Wit.Ai that was originally developed as a date parser, although it incorporates other useful capabilities. It transforms unstructured data into structured data that can be easily used by the chatbot. As an example it parses "*3rd of January of 1997*" into "1997-01-03T00:00:00:000-00:00". It is a hybrid model that uses both rules and machine learning to extract those entities. It is able to extract and parse a large group of entities, from which some will be used in this work. Namely, *time* will be very important for plan creation and visualisation, *number* may refer to an option, to a rating or even to personal information (weight or height), and *ordinal* is crucial for option selection. The main advantage of using *Duckling* is that it can recognise time intervals (such as "*weekend*" or "*the next couple of days*"), full dates (like the example given above) or even days of the week and parse them to the current time and date. This means that the user may indicate a day of the week without the need to introduce a date, for example. This makes the chatbot more aware of the current date and time and improves the quality of both the conversations and the queries.

5.1.7.3 Sentiment Analyser

The aim of including a Sentiment Analysis component was to further model the preferences regarding the plan. The sentiment captured in the input of the user is extracted by this component as an entity that fills up a slot of the agent's memory. The sentiment is usually negative, neutral or positive and can also have a magnitude value, which represents the intensity of the sentiment. This way it can be used by the actions to adapt the response and rate the meal elements. This component only requires the user's input text with no additional processing, besides the spellchecking that runs at the beginning of every new input. This component will be further explained and analysed in Section 5.2, that deals with preference modelling.

[6] http://wordlist.aspell.net/
[7] https://pyspellchecker.readthedocs.io/
[8] http://aspell.net/
[9] https://duckling.wit.ai/

5.1.7.4 Synonym Mapper

Rasa also enables the implementation of a Synonym mapper. This component is responsible for converting extracted entities into defined synonyms, allowing for a reduction in variability and normalisation. It requires a list of possible synonyms for a given entity value (see Figure 5.1). That being said, it can be another solution for misspelled words, as the most common mistakes may be included in the list. In this work, the synonym mapper will also work as a normaliser for plan-related entities. All the keywords in the database are all in upper case, hence the synonym mapper will convert entities such as "*dinner*" into "*DINNER*", so that this transformation does not have to happen in the custom actions functions.

```
## synonym:DINNER
- tonight
- night
- dinners
- dinner
- dinning
- dine
- dining
- diners
- this night
- diner
```

Figure 5.1: Synonyms defined for the word "DINNER"

5.1.8 NLU Final Pipeline

Considering the results explored in the previous sections, as well as the extra components that were described, the final pipeline consists of the following components:

1. *SymSpell* Spellchecker
2. VADER Sentiment Analyser (described in the next section)
3. *spaCy* Tokeniser
4. *GloVe* word vectors as dense features
5. Regular Expression features (sparse)
6. DIET Classifier
7. *Duckling* Entity extractor
8. Synonym Mapper

There are some important considerations regarding the pipeline. The order of it is of uttermost importance, since some of the components take as inputs, the output of previous ones.

The Spellchecker, the VADER sentiment analyser and the tokeniser require only the text input. On the other hand, all featurisers (GloVe vectors and Regex features) require tokens to work properly. DIET classifier needs the features previously extracted in order to perform both intent classification and entity recognition, whereas the Duckling Extractor only requires tokens. The synonym mapper must be the last component since it maps the recognised entities into new values according to the examples that were given in the training data. The user input is passed through all these components and the output is its inherent intent and the entities that are present.

A similar experimental evaluation (3 times with 80/20 train/test ratio) was performed for this pipeline and some extra metrics were extracted from the results. Table 5.14 resumes the weighted averages for the final pipeline. As it is possible to observe, the results are very similar to the ones obtained in Table 5.12.

Table 5.14: Intent classification and entity recognition results for the final pipeline

	Weighted Average			**Accuracy**
	Precision	**Recall**	**F1**	
Intent	0.899 ± 0.016	0.879 ± 0.020	0.890 ± 0.018	0.893 ± 0.018
Entity	0.902 ± 0.020	0.889 ± 0.019	0.890 ± 0.021	–

Furthermore, Figure 5.2 shows the number of correct and incorrect guesses according to the system's confidence. The results shown in this graphic are important for the selection of threshold values used to accept or reject a prediction, namely the NLU threshold mentioned in Section 4.3.2.3 for the Fallback Policy. In this case, a good threshold value would situate between 0.65 and 0.80, since this is the interval where the number of hits and misses are close to each other. The value should be chosen according to the metric which is to be optimised: a lower value would increase recall while a higher value would increase precision. In this work, the selected value was 0.65 in order to tackle the lower recall caused by the class imbalance of the dataset. Additionally, the confusion matrix for intent classification is shown in Figure A.4.1.

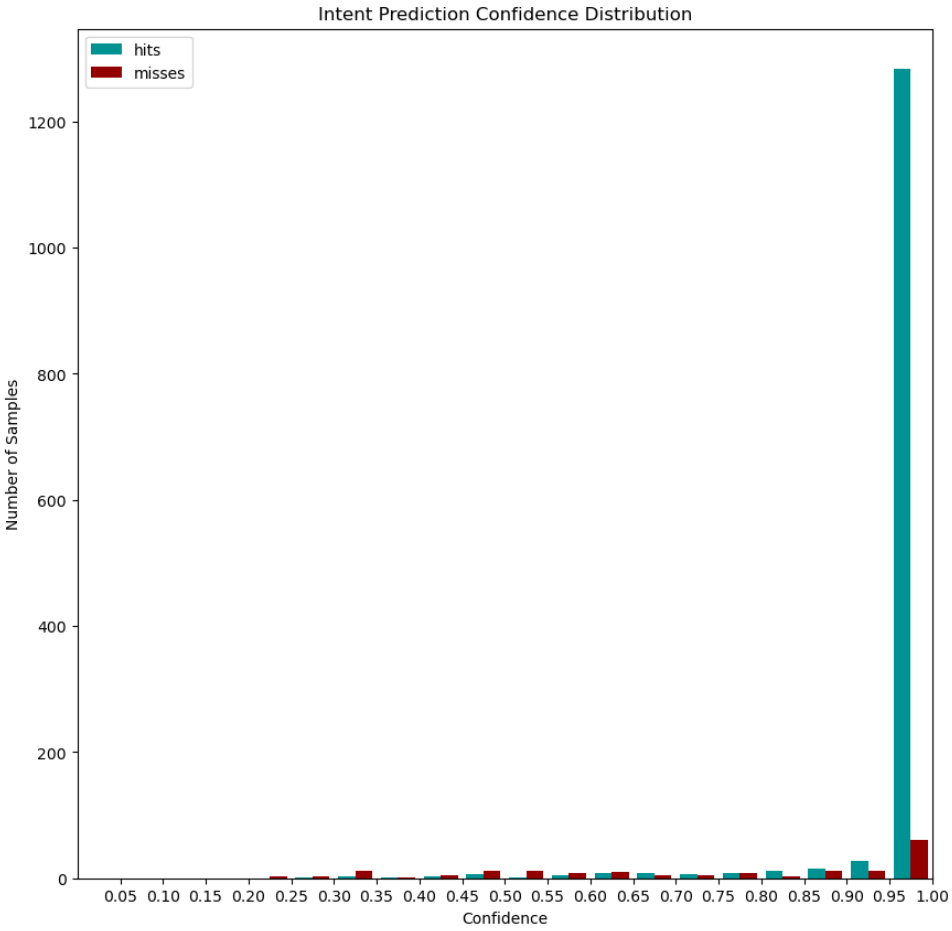


Figure 5.2: Histogram of hits and misses according to model confidence for intent classification

5.2 Preference Modelling

One of the advantages of utilising Natural Language as a means of communication is the increased facility of expressing preferences and opinions. However, humans have developed a preference model based on conversational data that can be a challenge for a computer to learn. According to Ohlsson et al. [183], food preferences correspond to the evaluative positions that are expressed by people toward foods, including the qualitative evaluation and also how much people like and dislike foods. Indeed, a machine will require a quantitative rating of foods in order to appropriately model the preference. However, the usual hedonic scale (*"How much do you like or dislike a food?"*) used to express the preference takes away the naturalness of the interaction through conversation. Therefore, the preference expression use case (UC.10 and UC.11 from Section 4.4.3) required the implementation of an algorithm capable of transforming the qualitative preference into a quantitative value. Sentiment mining from text may play a role in this task.

The quantification of the sentiment present in a sentence is usually related to the subject of the sentence. This means that, when expressing preference through verbal resources, the overall

sentiment of the input may be a good indicator of the acceptability level regarding a given food. Currently, there are a huge number of sentiment analysis tools that can be applied to any pipeline. In this work, two widespread sentiment analysers will be studied: VADER NLTK Sentiment Analyser[10] and Google Cloud Natural Language API sentiment mining component[11]. These two options were selected due to the fact that both produce a magnitude score regarding the intensity of the sentiment, which will be used to model the preferences.

In order to create a numeric model that is appropriate to the scope of this work and also validate the results, the data gathered during the collection of the dataset referred in Section 5.1.1 was used. As a reminder, volunteers were asked to give several examples of preference expression combined with the perceived preference intensity value from 0 to 5. As an example, the sentence *"I hate apples"* would have an intensity value closer to 0 than the sentence *"I don't like apples!"*, since the sentiment in the first is stronger than in the latter example. The following sections will explore the aforementioned tools and the results obtained.

5.2.1 VADER NLTK Sentiment Analyser

Valence Aware Dictionary and sEntiment Reasoner (VADER) is a lexicon and rule-based sentiment analysis tool based on the work of Hutto et al. [184] that was developed for sentiments expressed in social media. It is prepared to deal with slang and abbreviations that are very common in text messaging. This dictionary is sensitive both to the polarity and the intensity of sentiments. The VADER dictionary is used jointly with the NLTK library in order to process the sentences and extract a result for each. The dictionary has a set of positively and negatively rated tokens that are matched to the user sentence. In addition, Hutto et al. conducted an investigation on a large dataset of *tweets* to reach a group of heuristics that better outline the sentiments in sentence. The authors concluded that there were 5 heuristics that alter the flow of the sentiment in a sentence:

- Punctuation – The presence of punctuation, namely the exclamation point, on written text corresponds to an increase in the magnitude of the sentiment being expressed. A sentence that finishes with an exclamation is regarded as more intense than the same sentence but with a normal point.
- Capitalisation – Another interesting feature of informal written text is the capitalisation of some words in order to emphasise them. This is particularly clear when dealing with all-caps tokens that really maximise their intrinsic sentiment
- Degree Modifiers – Degree adverbs usually alter the intensity of the word to which they refer. This can represent a positive or a negative change, depending on the adverb.
- Contrastive Conjunctions – The use of contrastive conjunctions such as *"but"* or *"however"* can shift the polarity of the sentiment in the sentence. Normally the sentiment that follows the conjunction is the sentiment that prevails in the sentence.

[10] https://www.nltk.org/howto/sentiment.html
[11] https://cloud.google.com/natural-language

- Tri-gram analysis for sentiment flipping – The tri-gram rule was empirically defined as the optimum number of tokens to consider when checking negation of sentiment in a sentence.

VADER works by applying this set of rules to the tokens of the sentence and conceding a score to them. The score is then added and normalised between 1 and -1. This result corresponds to the compound score or magnitude of the sentiment in that sentence. This method also returns another value that represents the ration of parts of the sentence or document that fall in each polarity level: negative, neutral or positive. Under the scope of this work, the compound value will be used both to asses overall sentiment in the sentence and sentiment quantification. Moreover, VADER is very fast, which is a desired characteristic for a real-time agent, such as a chatbot.

5.2.2 Google Cloud Natural Language API

On the contrary, Google Cloud's sentiment analyser is a Deep Learning based approach. The algorithms and models are not public and can not be accessed nor is information about them available. However, it is known as one of the best NLP pipelines. Similarly to the VADER algorithm, it rates each sentence from -1 (negative) to 1 (positive) according to its sentiment intensity. This function has a cost associated that is dependent on the number of requests that it receives. Nonetheless, it is important to compare both models in order to understand its differences and limitations. It also has a mode that calculates the sentiment for a given entity, however after results visualisation, the standard mode for whole sentence sentiment was selected.

5.2.3 Sentiment Analysis for preference modelling

With a view to understand which classifier was the best to be incorporated in the conversational agent, both were tested against the ground-truth collected in the Google Form. The distribution of answers is depicted in Figure 5.3.

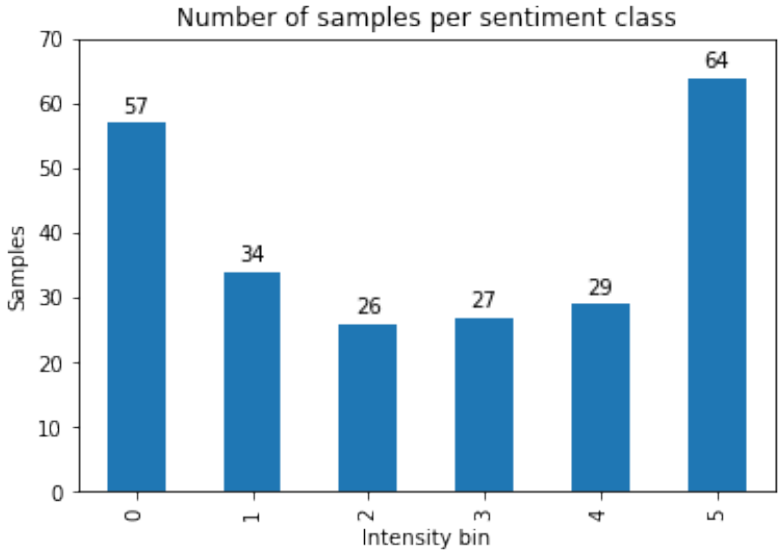


Figure 5.3: Distribution of samples per intensity class

The algorithms previously presented were tested using the data collected in order to create a model that can be adapted to the scope of the work. The sentiment magnitude of each sentence in the dataset (after the removal of out-of-scope samples) was calculated and stored. Both algorithms rate the intensity of the sentiment from -1 to 1, whereas the form asked for a preference rating ranging from 0 to 5. This interval is more appropriate for human understanding. For that reason, the scores stored from the algorithms were normalised to match the 0 to 5 range using a simple linear function, represented in Equation 5.1.

$$rate_{[1,5]} = (rate_{[-1,1]} + 1) \times \frac{5}{2} \tag{5.1}$$

After the transformation some metrics were analysed in order to assess the quality of the predictions, taking as ground truth the ratings collected in the form. The Spearman's correlation was calculated between the ground truth and the scores obtained by both models. This metric evaluates whether the relationship between two distributions can be described by a monotonous function and gives a score from -1 to 1, where -1 means that while one decreases the other increases and 1 means that they follow the same direction. Simply put, it is 1 if all the samples in both distributions are in the same ordinal place. The score is then penalised according to the difference in order. The Pearson correlation was also calculated. It is similar to the Spearman's correlation, although it is less flexible to outliers since it tries to relate the two distributions using a linear function.

Moreover, error metrics (Mean Absolute Error and Root Mean Squared Error) were calculated for the whole distribution as well as per class. Additionally, statistical metrics (Mean, Median and Standard Deviation) were calculated per class. The results can be observed in Tables 5.15 and 5.16.

Table 5.15: Spearman and Person correlation, Mean Absolute Error and Root Mean Squared Error for the complete dataset, per sentiment analyser

	Google Cloud	**VADER**
MAE	0.85	1.11
RMSE	1.20	1.45
Spearman	0.73	0.67
Pearson	0.82	0.68

Table 5.16: Measures of central tendency per class, for each classifier (both capped and not capped)

		Google Cloud	VADER	Google Cloud capped	VADER capped
0	Mean	0.38	1.68	0.07	1.19
	Median	0	1	0	0
	Std. Dev	0.56	0.85	0.37	1.34
1	Mean	0.21	1.68	0.12	1.24
	Median	0	2	0	2
	Std. Dev	0.73	0.73	0.69	1.18
2	Mean	0.73	2.15	0.46	2.04
	Median	0	2	0	2
	Std. Dev	0.96	0.73	0.99	0.96
3	Mean	2.93	2.81	2.89	2.89
	Median	3	3	3	3
	Std. Dev	1.82	0.92	1.95	1.19
4	Mean	4.90	3.76	4.90	4.14
	Median	5	4	5	4
	Std. Dev	0.41	0.58	0.41	0.88
5	Mean	4.23	3.42	4.22	3.95
	Median	5	4	5	5
	Std. Dev	1.56	1.04	1.68	1.61

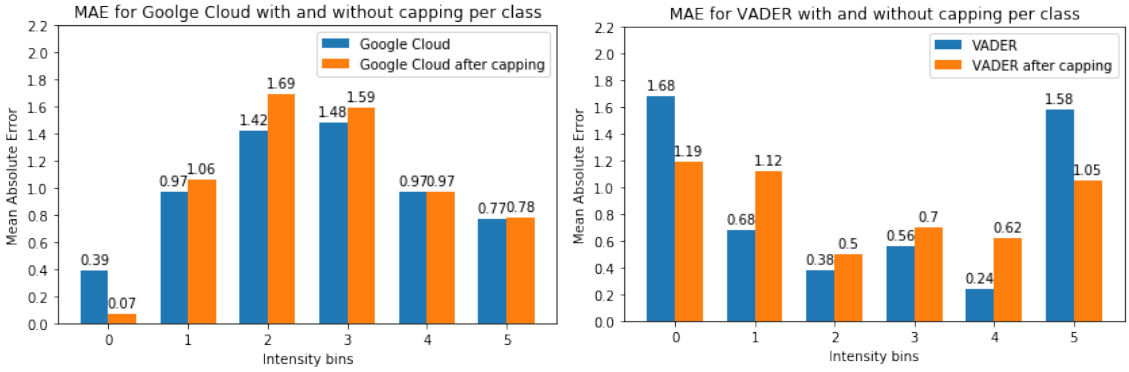
At a first glance, it is possible to notice that the Google Cloud Spearman's correlation value is higher than VADER's. This means that the model produces predictions that align better with the ground truth than the values predicted by VADER. However, both values are well above 0 and close to 1, which shows that both can correctly scale the intensity of the sentiments when expressing preferences. The same happens to Pearson's correlation values, meaning that, once again, both models can predict sentiment intensity variation over that scale.

Looking at the error values, it is possible to see that the Mean Absolute Error was lower for the Google model than for VADER. After some further analysis to understand the reason behind these values, it was possible to conclude that the Google Cloud model dealt better with outliers and failed attempts than VADER. This is specially clear in the most extreme classes (0 and 5). These classes also produce the largest errors when the model fails to predict or gives a wrong prediction since they are in the extremes. Apart from this, the classes are not balanced in the dataset. Indeed the classes that have the most samples are exactly 0 and 5 (Figure 5.3).

This fact may be biasing the mean error metrics for the VADER classifier. Therefore, another approach was taken to analyse the data. The error metrics were calculated per class, see Table 5.17 and Figure 5.4.

Table 5.17: Average values of MAE calculated by class for both models before and after capping

	Google Cloud	**VADER**	**Google Cloud capped**	**VADER capped**
MAE	1.00	0.85	1.03	0.86



(a) MAE per class for Google Cloud's sentiment Analyser (capped and no capped)

(b) MAE per class for VADER's sentiment Analyser (capped and no capped)

Figure 5.4: Values of Mean Absolute Error calculated per class for both models

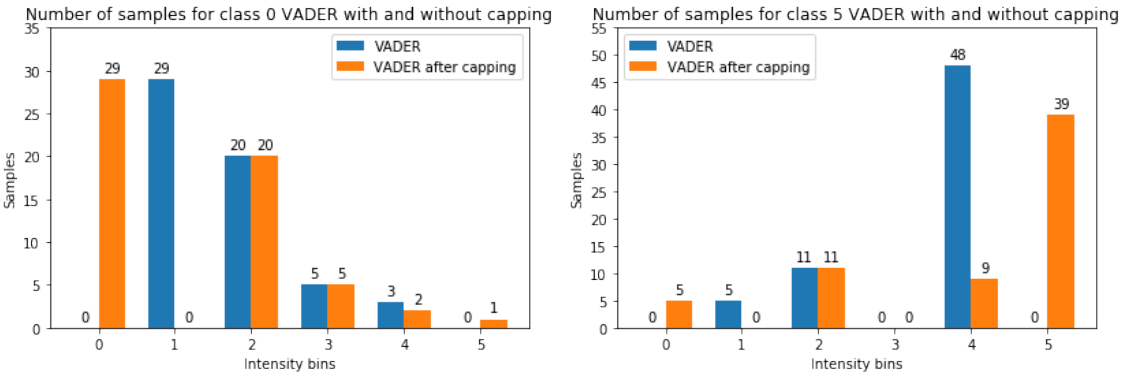
Regarding VADER, it confirmed that the largest error values happen at the extremities. Also, by looking at central statistical values per class, namely median, the pattern repeats (Table 5.16, second column). The largest differences from the prediction median and the value of the class happen at the extremities. This is explained by the fact that there are no more classes before 0 and after 5, concentrating all the possible values in the next bin. This phenomenon did not allow the lower values to compensate for the higher ones and vice versa. Also, it is very hard to have a prediction exactly at the extremes since it would represent the most negative or positive sentiment that one can express using text. In this case, the feeling towards an ingredient, even if the user does not want that ingredient in the meal plans, will never reach those levels. On the contrary, the predictions for the classes 1 to 4 have central values that are closer to the real value, also having a lower standard deviation. For these intervals, the model actually has a good predicting power. Mean absolute error values are also lower in these central classes.

On the other hand, Google Cloud sentiment analyser shows a completely opposite pattern. The error values are actually lower in the extremities and higher in the centre. When looking at median values (Table 5.16, first column), it is possible to notice that, although the values for 0 and 5 are closer to the correct value, the values from the centre do not match the correct bins by a large difference.

The final model to express preferences has to take into account the Lifana recommendation system rating mechanism that ranges from 0 to 1. The extreme values of the rating scale in the Lifana engine have special characteristics, namely the negative or zero-valued rating. If any recipe or ingredient get a zero rating, the recommender will stop including that element in future meal plans. Bearing in mind that it is virtually impossible to have a score of zero using either sentiment analyser, a solution that simply maps the scores from the [-1; 1] range to [0; 1] range would not consider this. For that reason, an inferior and superior thresholds were created to map some predictions to either zero (below the inferior threshold) or five (above the superior threshold). The prediction values between those thresholds were preserved and mapped linearly to the correct scale. This way, it is possible to have the *remove* action by expressing a very negative sentiment. Taking this into account, and also the fact that Google Cloud is not free for a large scale, VADER was selected as the sentiment analysis method used to model the preferences

In order to precisely create the model, the inferior threshold was set on the value of the median corresponding to the predictions for the 0 class. The same logic applies to the superior threshold, it was set on the median value of the predictions for the 5 class. Thus, the preference model follows the algorithm expressed in Equation 5.2. The distribution of predictions for both the 0 class and the 5 class, before and after capping by the median value, for each one of the models are represented in Figures 5.5 and 5.6

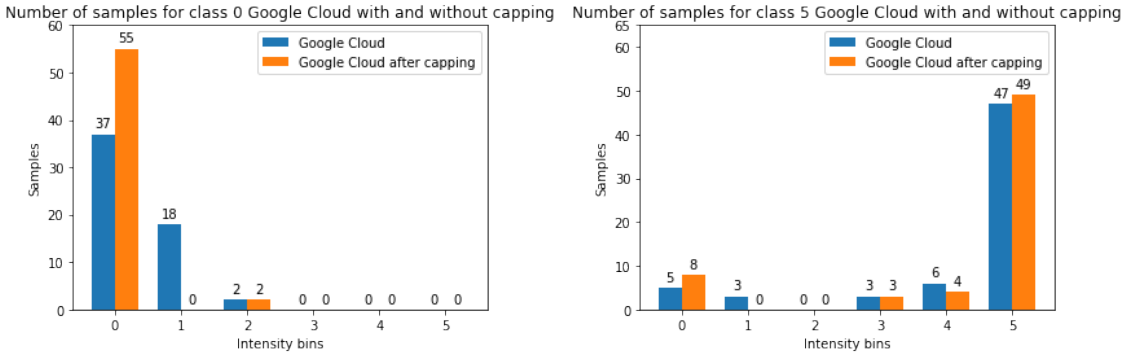
$$preference = \begin{cases} 0 & prediction \leq -0.60 \\ \frac{prediction+1}{2} & -0.60 \leq prediction \leq 0.60 \\ 1 & 0.60 \leq prediction \end{cases} \tag{5.2}$$



(a) Number of predictions for class 0 using VADER capped and no capped

(b) Number of predictions for class 5 using VADER capped and no capped

Figure 5.5: Difference in predictions for VADER's model before and after applying the capping rule described in Equation 5.2, for classes 0 and 5



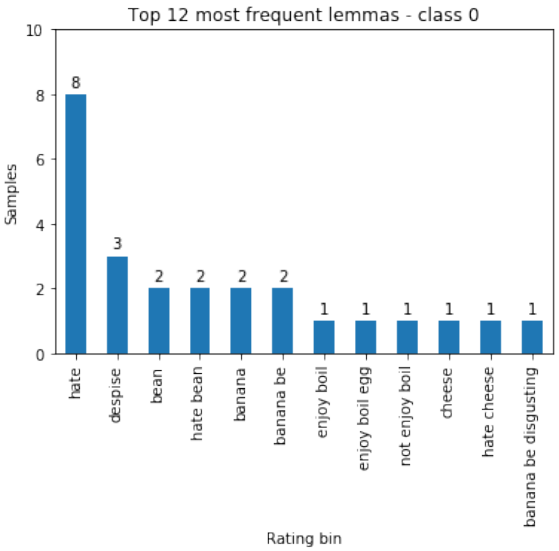
(a) Number of predictions for class 0 using Google Cloud model capped and no capped

(b) Number of predictions for class 5 using Google Cloud model capped and no capped

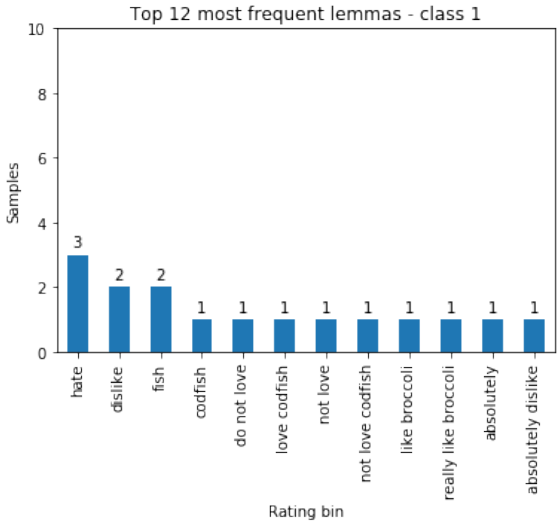
Figure 5.6: Difference in predictions for Google Cloud's model before and after applying the capping rule described in Equation 5.2, for classes 0 and 5

It is noticeable the change in the number of predictions rated 0 and 5, before and after applying the capping rule. The change is bigger in the VADER classifier than in Google Cloud's model, as expected. However, looking at Table 5.16, it is possible to note that the median and mean values for each class are more accurate using VADER's model then Google Cloud's model, after capping the results.

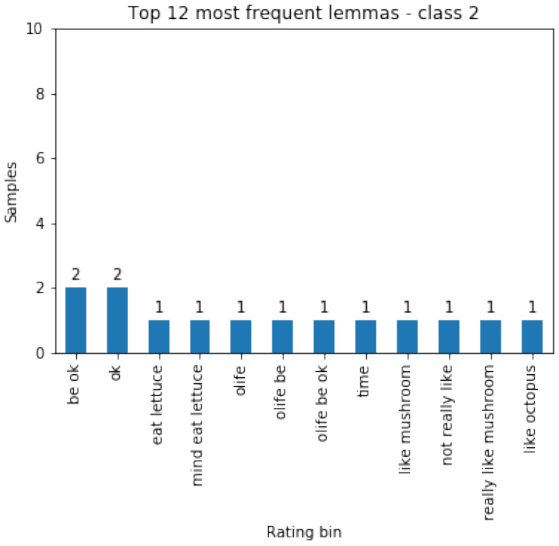
In addition, if the rating falls inside the central interval (from 1 to 4), the chatbot will save the correspondent preference value (normalised to [0;1]) and ask the user for a rating from 0 to 5, in order to confirm the perceived rating. This would cover for failed predictions and the increased variability present in the middle classes. As a matter of fact, by examining the most frequent lemmas for each class, this pattern became evident. As it may be perceived in the graphics of Figure 5.7, the variability of word lemmas in the central classes is higher than in the extremities. The latter have higher homogeneity in terms used. Nevertheless, if the user is not willing to give a rating, the predicted evaluation will still be stored, that is the reason why it is crucial to have high accuracy regarding the central classes.



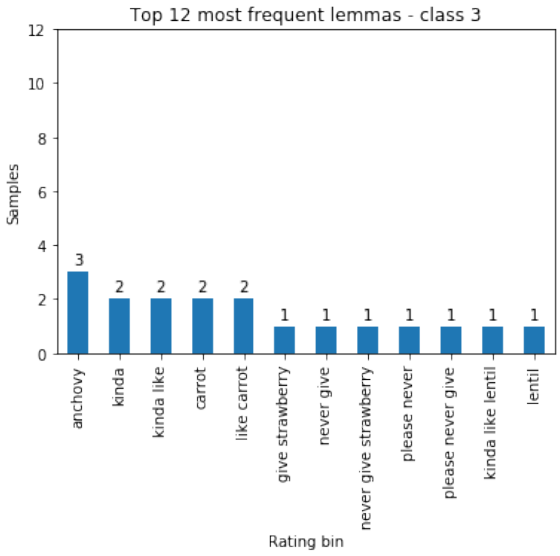
(a) Top 12 most frequent word for rating intensity 0



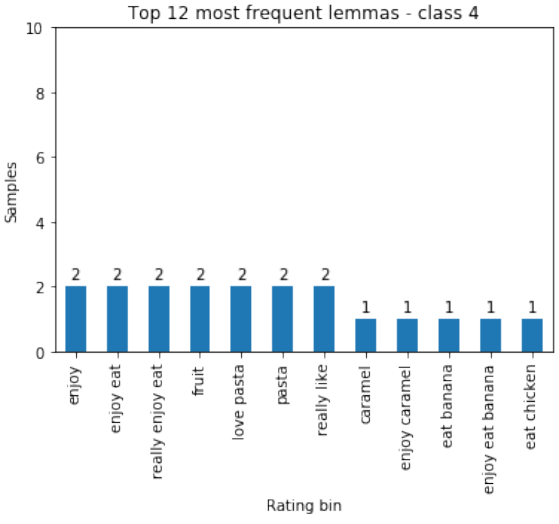
(b) Top 12 most frequent word for rating intensity 1



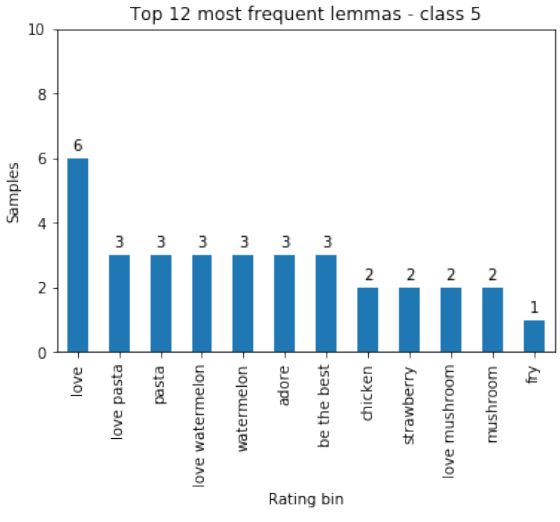
(c) Top 12 most frequent word for rating intensity 2



(d) Top 12 most frequent word for rating intensity 3



(e) Top 12 most frequent word for rating intensity 4



(f) Top 12 most frequent word for rating intensity 5

Figure 5.7: The plots show the top 12 most frequent word lemma for each class. It is possible to see that the central classes (1 to 4) have increased variability. On the contrary, classes 0 and 5 are more heterogeneous, having some terms that appear much more than the others.

5.3 Food matching

Another crucial challenge that appeared while trying to model dietary preferences supported by a dataset of ingredients and recipes was the matching algorithm that mapped the entities captured by the NLU module from the user sentence, to the correct entries in the dataset.

The first approach to solve this problem consisted in performing a word-based search in the database for exact matches on ingredient names. The method was improved with a fuzzy word matching to acquaint for word variations. Although there is not a high variability in the names of ingredients, this method often left several correct options unretrieved and was not taking into account food groups. In fact, people generally express their preferences towards broad groups of ingredients whose name does not match the name of the ingredients. This means that when considering more than one ingredient that can be grouped together, people often use their hypernym to express a preference. Following this line of thought, the implemented algorithm should be capable of discerning between individual ingredients and food groups. This lead to the exploration of synsets and lexicon databases such as *WordNet*[12] [185] or *ConceptNet*[13] [186] as way to classify each ingredient. Furthermore, the combination of these knowledge bases with the information provided by the *LanguaL* facets on each ingredient created an interesting set of features to classify the ingredients. However, there was still the challenge of connecting the ingredient to the respective representations on the dataset. This is considered as a ranking task in the field of information retrieval. Based on the work of Wu et al. [187], the approach designed for this task is based on using or creating embeddings for ingredient classification. The logic behind this method is that any entity can be embedded by a neural embedding model by learning feature representations relationships among collections of those entities. The vector space is the same for all entities, which enables the model to rank entities, documents or objects according to the similarity measure to a give query entity. The latter is not required to be of the same type as the items being ranked. In order to create proper embeddings, two paths could be taken: provide annotated data to the model, such as query entity, database matches tuples; or learning unsupervised embeddings from the relationships between the entities. Bearing this in mind, *LanguaL* facets may pose as classes for the ingredients, namely the facets A, B and C, which will be further explained (the full list of categories used is in Table A.3)

- Facet A – The ingredients are classified according to a food group to which they belong. In this case, facet A gathers several international standards for food grouping. For the purpose of this work, the classification from the European Food Groups will be considered since it was regarded as the one with the most granularity.
- Facet B – It addresses the food source and has several levels of hierarchy. For this work, only the last and more specific level will be considered. As an example, *milk*'s food source is *cow* and *raisin*'s food source is *grape*. This facet is particularly important to aggregate foods that correspond to their food source, such as fruits and vegetables or types of fish.
- Facet C – The last one categorises the part of the animal or plant from which the ingredient is extracted. To illustrate, the descriptor for *cheese* under this classification is *milk*. This

[12] https://wordnet.princeton.edu/
[13] https://conceptnet.io/

facet presents the least connection to current language terms, although it can be important to discern from similar ingredients semantically.

A correct classification of the ingredient into these three groups enables the retrieval of possible candidates. Still and all, given the available data none of the aforementioned paths was an ideal option for good performance, since some of the groups have only one ingredient. This will not generalise well for new data.

Given this, the idea of using embeddings to map both the ingredients and their facets into a vector space required the resort to pre-trained vectors. Howbeit, most of the pre-trained word embeddings available, like Word2Vec or GloVe, were generated by processing huge corpora of generic text, with no special attention for the food lexicon. This fact may harm the performance of the food matching algorithm. The work developed by Jaan Altosaar[14] generated a set of pre-trained embeddings focused on food, using a corpus of recipe instructions. The goal of his job was to create a recipe recommendation system that joined ingredients in order to create new recipes based on the embeddings of those ingredients. Although the embeddings are specialised for food, they do not capture the semantic relationships between hypernyms and hyponyms among ingredients. Instead, these embeddings encoded the relations that several ingredients have when used together. As an example, according to this methodology, *parmesan* is closer to *pasta* than to *cheddar*, even though they are two types of cheese. This is due to the fact that the first two are used together many times in recipes whereas it is rather rare to mix *parmesan* and *cheddar* in the same recipe. This approach would not be appropriate for ingredient classification, although it is an interesting approach to new recipes and can boost their creation for the Lifana recommender system database. A similar approach is followed by Tansey et al. [188] which encodes complete diets into a vector space using a combination of Word2Vec embeddings and nutritional information.

5.3.1 ConceptNet Numberbatch and Retrofitting

Both approaches previously referred prove that food information may be encoded in a vector space. Bearing this in mind, a set of pre-trained embeddings generated from the ConceptNet knowledge graph was regarded as a suitable option to encode the ingredients and the *LanguaL* descriptors into a vector space while maintaining the semantic relations between terms. *ConceptNet Numberbatch*[15], as stated before, is a pre-trained set of embeddings that take into account the structured common sense knowledge from ConceptNet, which grants information about their meanings besides the context in which they appear. It was built using an ensemble that joins information from ConceptNet, GloVe, Word2Vec and OpenSubtitles 2016 [186], employing a technique called retrofitting which will be further explained.

Retrofitting is a technique developed by Faruqui et al. [189] that aims at incorporating the data present in semantic lexicons such as WordNet or ConceptNet into a previously defined word vector space. This method does not generate new embeddings, instead it refines the vector space to account for relational information, meaning that words which are lexically linked

[14]https://jaan.io/food2vec-augmented-cooking-machine-intelligence/
[15]https://github.com/commonsense/conceptnet-numberbatch

together should have similar vector representations. It works by applying a linear vector transformation to the vectors that closes the gap between related words and increases the distance between lexically unrelated words. The transformation leads to a loss function Ψ that should be minimised and it is represented in Equation 5.3, where $\hat{q}_i$ is the initial vector, q_i the retrofitted vector and q_j its neighbours in the ontology (the vectors that are supposed to get closer). The parameters α and β control the relative strength of each parcel in the equation and can be tuned for maximum performance.

$$\Psi = \sum_{i=1}^{n} \left[\alpha_i \left\| q_i - \hat{q}_i \right\|^2 + \sum_{(i,j) \in E} \beta_{ij} \left\| q_i - q_j \right\|^2 \right] \tag{5.3}$$

In order to minimise the loss represented in Equation 5.3, it must be differentiated, resulting in Equation 5.4, which correspond to the linear transformation applied to the vectors.

$$q_i = \frac{\sum_{j:(i,j) \in E} \beta_{ij} q_j + \alpha_i \hat{q}_i}{\sum_{j:(i,j) \in E} \beta_{ij} + \alpha_i} \tag{5.4}$$

By carefully analysing the equation, one can conclude that it corresponds to the weighted average of the initial vector embedding and the vectors representing the concepts that are linked to it, controlled by parameters α and β, where the first attributes increased relevance to the initial vector and the latter controls the importance of the linked concepts.

5.3.2 Experimental Evaluation

With a view to create a mechanism for ingredient classification, Numberbatch's pre-trained embeddings were used to encode the information. Each one of the aforementioned *LanguaL* facets were encoded using the average of the embeddings of each word composing the name of the facet, after stop-words removal using NLTK's dictionary of stop-words. The names are usually composed by more than one word so this process required several steps:

1. Tokenisation – In order to obtain each individual token, spaCy tokeniser was used
2. Stop-words removal – Each extracted token is compared with a dictionary of stop-words so that they are removed to prevent retrofitting with them
3. Word normalisation – Words are normalised in order to remove plural inflections which is particularly common in this dataset
4. Dictionary matching – The last step concerns matching the tokens with the words in the dictionary to extract the embeddings. However, there are some words that are not present in the lexicon or are not in the correct format. To solve this, the following steps are performed:
   (a) Create all possible bi-grams and verify if they exist in the lexicon
   (b) If not, verify if the individual tokens exist in the lexicon

(c) If not, check whether any tokens have hyphens (also very common) and remove it by first creating a bi-gram and if the bi-gram is not part of the lexicon join the tokens. The last step here is to address each part as one separate token and check if it is present in the dictionary

(d) If not, perform fuzzy matching using *FuzzyWuzzy* Python library. It calculates Levenshtein distance to all possible words in the dictionary and retrieves the most probable candidate that has only one edit of distance, if there is any

(e) If not, create a zero-valued vector to emulate the embedding.

Some of the rules are based on data exploration by finding patterns and common aspects that can influence the process. For example, regarding the group *Fish and Seafood*, the resulting embedding would be the average of the vectors for *Fish* and *Seafood*, ignoring the word *and*. Using this method it was possible to classify each ingredient as one group from each facet. The information about the distribution of classes is present in Table 5.18. Note that not every ingredient have all three of the descriptors.

Table 5.18: Details about ingredient data

	facet A	**facet B**	**facet C**
Number of classes	33	213	79
Number of ingredients	903	902	902

After creating embeddings for each ingredient and descriptor, classification was done and in order to assess the quality of the results, accuracy was taken per class. Table 5.19 resumes the results obtained for two sets of pre-trained embeddings: GloVe (no ontology information) and ConceptNet Numberbatch (retrofitted with ontology information).

Table 5.19: Accuracy results for ingredient classification according to *LanguaL* facets, using several different pre-trained embeddings models

Model	**Accuracy**		
	Facet A	**Facet B**	**Facet C**
GloVe 400k vocabulary	0.28	0.32	0.09
ConceptNet Numberbatch	0.42	0.40	0.09

The classification was made according to the same criterion used by Wu et al. in [187], which is the cosine similarity between the embeddings of each ingredient and the embeddings of the class. The most similar class (higher value) was regarded as the prediction made by the

model. As it possible to note, the results obtained by the Numberbatch approach were higher than the ones obtained using GloVe. This proves that retrofitting actually incorporated some semantic information in the embeddings.

However, the results were not very prominent and left a large margin for improvement. For that reason, being *LanguaL* an ontology which incorporates lexical information about food in a knowledge graph style, these established relations may be used to retrofit the pre-trained Numberbatch's embeddings so that they become more aware of food semantics. It is expected that this technique would improve the results attained before retrofitting. Several experiments were conducted in order to understand the best way of creating an appropriate embedding dictionary. There were 3 possible ways of performing retrofitting with the available data:

A. Retrofitting class embeddings with the vectors from the ingredients that belong to each class

B. Retrofitting ingredient embeddings using the vectors of the classes they belong to

C. Combining the two previous approaches and retrofit both the ingredients and the classes

Each procedure presents advantages and disadvantages. Procedure **A** will not change the embeddings associated to each ingredient, instead it encodes each label according to the ingredients that it contains. This may pose as an advantage for real world applications when dealing with ingredients that are not part of the database. In these cases, provided that there are similar ingredients in the database, the model would still be able to classify the query and handle the preference. Contrarily, procedure **B** changes the vectors that represent the ingredients in order to match the class label embedding. It is expected that this method will converge easily since each ingredient will only be retrofitted using, at the most, 3 concepts, which does not happen in the former method. Procedure **C** tries to change both elements towards a converging representation. This will change adapt both data types to each other which may be harmful when handling new information. Also, convergence may not be achieved since some of the concepts are related in different ways. The selected approach should consider the performance obtained after retrofitting for all three classes at the same time, since there will only be one embedding representation per ingredient. Retrofitting one class individually may harm previously retrofitted classes. Even though one method may provide better results than other for an individual facet, the joint performance should guide development. The results were obtained using K-Fold Cross Validation with 6 randomly selected folds (see Table 5.20). This would assure the methodologies were properly validated. The retrofitting session lasted for 20 iterations and after each iteration the results are validated. According to the results, the β parameter is updated. If the results increased overall (among the 3 facets) the value is not updated and runs the next iteration with the same weight, otherwise, the value is increased by a factor of 20%. The initial β value is set 1, the same as α, however, the latter is fixed at 1 for the whole session. This way, at the starting point, both the vector to be retrofitted and the definitions have the same weight, which allows not to lose intrinsic information about the words.

Table 5.20: Accuracy results for ingredient classification according to *LanguaL* facets, using three different retrofitting approaches

Approach	**Accuracy**		
	Facet A	**Facet B**	**Facet C**
Baseline results (no retrofitting)	0.412 ± 0.044	0.447 ± 0.027	0.108 ± 0.023
Approach A	0.6475 ± 0.029	0.505 ± 0.050	0.402 ± 0.046
Approach B	0.412 ± 0.044	0.447 ± 0.027	0.108 ± 0.023
Approach C	0.504 ± 0.054	0.417 ± 0.050	0.242 ± 0.035

By looking at Table 5.20 it is clear that the approach that produces the best overall results is approach A, where the classes were retrofitted incorporating information about the ingredients. This proved the hypothesis that approach A deals better with unseen data than the other approaches, by not altering the values of the ingredients. During the training of approach B it was not possible to make the model converge. This makes sense when thinking about the testing process. The ingredient embeddings of the training set are being altered according to the *LanguaL* information, however, the ones in the validation set have not suffered this alteration. This means that the classification score will be mostly the same in this case. Regarding approach C, it is possible to see an improve in both facets A and C although facet B accuracy decreased. Bearing in mind these results and the perceived good handling of new data, approach A was selected as the method to create the new set of embeddings that will be used in ingredient retrieval.

Despite the fact that there is a clear improvement in classification accuracy, due to the naming format of the ingredients, the embeddings may be considering information that is not relevant for classification. The list below illustrates some examples of ingredient names present in Lifana database.

- Pineapple, canned in juice
- Pasta, plain, fresh, raw
- Eggs, chicken, whole, raw
- Onions, raw
- Tuna, canned in brine, drained
- Cheese, Edam
- Peppers, capsicum, green, boiled in salted water

In most cases, the name includes extra information that may not be relevant for preference modelling, such as the cooking method or the way of preservation. Moreover, names do not always follow the same logic because they derive from several sources. As a consequence, it was not possible to create rules for name processing before retrofitting. An example of such a rule would be to remove every word after the first comma, however, as it noticeable in the examples above, some important characteristics regarding the ingredients are present after the first or even second comma. This pre-processing would have to be hand-made, which would be impractical. As a way to deal with this problem, an altered version of the TF-IDF algorithm (see Section 3.1.4.2) was applied as a weighting mechanism. This way, words that are not important to distinguish classes will have reduced importance when retrofitting the embeddings.

TF-IDF weights were calculated in two different ways, depending on what was considered a document. Before detailing the methods, it is important to refer that the element to be retrofitted and the items used to retrofit from now on will be referred as **concepts** and **definitions**, respectively.

1. Document corresponds to the definitions of each concept

   In order to illustrate this case, when retrofitting the concept *Fish and Seafood* a document would comprise all ingredient names that belong to that class. This would boost the Term-Frequency part since there are usually several ingredients with similar names, varying only the cooking method, for example. Also, this would punish words that appear in different concepts, such as the cooking or preservation methods that are common to different types.

2. Document corresponds to each individual ingredient/class name

   In this case, Term-Frequency will not benefit, although Inverse Document Frequency will punish even more the tokens that appear in many ingredients or classes

Embeddings were retrofitted once again using method A with each of the TD-IDF approaches and an extra hybrid one that uses Term Frequency calculated through procedure 1 and Inverse Document Frequency calculated using procedure 2. With this hybrid approach the goal is to further punish words that appear in many concepts while boosting words that belong to only one class and may be more important for a good performance. The results are presented in Table 5.21. Once again, they were taken using K-Fold Cross Validation with 6 folds.

Table 5.21: Accuracy results for ingredient classification according to *LanguaL* facets, using approach A for retrofitting with different methods of TF-IDF weighting

TF-IDF approach	**Accuracy**		
	Facet A	**Facet B**	**Facet C**
Baseline results (no TF-IDF weighting)	0.6475 ± 0.029	0.505 ± 0.050	0.402 ± 0.046
TF-IDF 1	0.690 ± 0.031	0.595 ± 0.064	0.451 ± 0.031
TF-IDF 2	0.682 ± 0.041	0.563 ± 0.061	0.411 ± 0.047
TF-IDF 3 (hybrid)	0.692 ± 0.026	0.595 ± 0.048	0.451 ± 0.021

In fact, TF-IDF weighting improves the results, as shown in Table 5.21. Every experimented method increased performance when comparing to retrofitting without weighting. However, it is clear that TF-IDF 3, the hybrid method, actually presents the best results overall showing improvements in all three groups of labels. Approach 1 presented similar results for facet B and facet C, although the error values are higher and the accuracy for facet A is lower. TF-IDF 2 presented the worst results of the three. The hybrid weighting method actually proved to be the best to deal with this kind of data due to selectively punishing the terms according to their frequency in different groups.

The evident improve in the results from using generic pre-trained embeddings assures that food semantic information available in *LanguaL* was incorporated into word vectors. The next section will explore the algorithm designed to classify and retrieve ingredients based on a query entity, which takes advantage of a set of pre-trained ConceptNet Numberbatch embeddings retrofitted with *LanguaL* semantic information, through approach A and using the hybrid TF-IDF weighting (TF-IDF 3).

5.3.2.1 Ingredient Retrieval

The purpose of creating word embeddings that captured the semantic relations present in the *LanguaL* ontology was to retrieve from the dataset the correct ingredients, given a query entity extracted from user input. A perfect retrieval process would gather all ingredients that correspond to the query based on the classification of that query. The entity may indicate to a group of ingredients, to a specific ingredient or even to a group of ingredients that do not match exactly the *LanguaL* labels described in the beginning of this section.

Bearing this in mind, the algorithm has the following steps:

1. Entity Recognition

The entity is extracted from user input through the NLU module described in Section 5.1.8

2. Pre-processing

   An identical pre-processing method to the one described in Section 5.3.2 is applied to the extracted entity, so that it matches the correct format of the embedding dictionary

3. Encoding

   The tokens or token can now be encoded into an embedding by averaging the embeddings of each token in the query

4. Classification

   The query embedding is then classified in each one of the three facets using cosine similarity as distance metric to rank the possible candidate classes. The class that presented the lowest distance value (highest cosine similarity) to the query entity is selected as the predicted label. That being said, each query is labelled with one *LanguaL* code per facet

5. First extraction process

   The algorithm will extract every ingredient from the database (through a REST request to the API) that matches at least 2 of the 3 *LanguaL* descriptors as possible candidates. These candidates are filtered according to two metrics of similarity based on different characteristics:

   - Fuzzy Word matching to find words that effectively match the query entity
   - Cosine Similarity for further filtering

   These two filtering modes will generate two lists of strong candidates.

6. Matching algorithm

   The mentioned lists are intersected in order to extract exact matches. If the intersection results in one element, that element is considered to be the correct match and the process ends. Howbeit, if the intersection results in more than one ingredient, further selection steps are followed. The premise is that the user would be probably referring to a group of ingredients (such as *meat*) or to an ingredient that has many different entries in the database (same food product with different cooking methods or preservation techniques). In this case, the algorithm will check if there is a strong match between the query and any of the *LanguaL* labels that classified it using cosine similarity measure from embeddings. If any of the descriptors have a similarity value superior to a defined threshold value, it is then matched using fuzzy matching. If Levenshtein distance is not superior to one edit, it is regarded as a match and the algorithm returns all the ingredients that are described using that *LanguaL* code. In case there are more than one match, it will return according to the order A, B, C. If none of the *LanguaL* descriptors pass the matching steps, two paths may be taken:

   - If the list of ingredients extracted through embedding similarity in step 5 is not empty, it will be returned as the final query result, being considered a group of ingredients that not match specifically any *LanguaL* descriptors

- If the list is empty, the query is regarded as having no matches in the database

The thresholds referred in the pipeline were defined through observation, in order to maximise accuracy. An increase in the value of the threshold would represent an increase in precision with a consequently decrease in recall. This method will be validated by user testing explained in Chapter 6, since there is a lack of an annotated dataset that could serve as validation.

5.4 Dialogue Management

In this section the procedures used to tackle the dialogue management part of the agent will be explored and analysed along with the obtained results for the different methods.

5.4.1 Responses

As mentioned in Section 4.3.2.4, responses can be retrieved from a set of pre-defined ones, usually in a text format (although they can be images or buttons) or custom actions that run a given code. The former are saved in the domain file, enabling the retrieval process. Moreover, the text responses can be triggered from inside custom functions in order to give feedback to the user. There are a total of 102 textual response samples (illustrated in Figure 5.8), divided by 57 response classes. It is possible to give more than one example for response, so that the chatbot presents variability in the answers it sends. These samples were created manually. The following section will explore the custom actions.

```
responses:
  utter_all_meals_changed:
  - text: All the meals were changed!
  - text: It is done! I have changed them!
  utter_anything_else:
  - text: Do you need anything else from me?
  - text: Is there anything else I can do for you?
  - text: Is there anything else I can help you with?
  utter_ask_activityLevel:
  - text: 'I need to know your activity level. Can you tell me?  Please use one of
      the following categories: lowest (never), low (once a week), medium (3/4 times
      a week) or high (everyday).'
```

Figure 5.8: Textual responses examples

Furthermore, in order to become a proper virtual assistant, the chatbot should be able to interact with external services. In this case, the agent will interact with the Lifana API, described in Section 4.1.1, through the custom actions. A total of 27 actions, including one form (see *Form Policy* in Section 4.3.2.3), were created using Python to map all the previously defined use cases. These actions can be grouped together since many of them are nested and are triggered from inside other actions. It is also important to mention that the authentication request to the API is coded to refresh the access token each 5 minutes, in order to prevent loss of authorisation and further complications with the request to the API.

5.4.1.1 Start Session

Rasa gives the ability to alter the action that runs in each session start. When overriding this action, it is possible to make it retrieve user information from the database through an API request and save it in slots. This information correspond to personal details from the user that may be useful for a more personalised experience. Moreover, depending on the retrieved information, this first action will trigger different responses in the form of a message. In case the user is already registered, the bot will send a greeting message that includes the user's name and some suggestions to commands that it can address. Otherwise, the bot sends a welcoming message including the same suggestions and, in addition, it asks whether the user is able to answer to some questions that will collect its personal information. After the variable welcoming message, the bot waits for user input so that in can proceed.

5.4.1.2 Initial Form

The chatbot was planned to work from end-to-end including the profile creation process. Therefore, a Rasa Form was developed to collect information in an organised and sequential way. It is triggered after a positive answer to the question mentioned in the previous section. The agent starts to ask for each detail (name, birth date, gender, among others). It then validate the information extracted from intents, entities or text itself and saves it in slots. For example, in the name question, the NLU module should extract the user's name as an entity (person_name) which is then saved in the slot corresponding to the name. If validation fails or the NLU module is not capable of extracting a correct intent/entity, the bot sends a message saying the input was invalid and asks for the information once again. This action is illustrated in Figures B.1.2 and B.1.3 and corresponds to the Use Case UC.12.

5.4.1.3 Create Plan

One of the core actions of the chatbot is the creation of plans (use case UC.1). This action is very complex and contains several heuristics to work properly. Plan creation is based on a time period. As mentioned in Section 4.1.1, this can be done either on a week basis and on daily basis. The first step of this action is the transformation of the period referred in the user input to a format that is compatible with the API. That being said, the *period* entity extracted from user input will be transformed into one of these four options: "now", "day", "week" or "month". If no *period* entity is captured or inputted, the action will verify whether a *time* entity was extracted or not. If the entity *time* was extracted from the input, the *period* is set to "day", otherwise it is set to "now" and the *time* is set to the date of that moment. Contrarily, if *period* is defined, a set of conditions are checked sequentially:

1. If "month" is part of the entity, *period* is set to "month". This will cover cases where entity is either "month" or "monthly". In this case, the chatbot will send a message warning the user for the extended time that creation process will require. Also, the message says that the system is not intended for monthly plans, since the nutritional content is considered for each week.

2. If "weekend" is part of the entity, *period* is set to "day" since it will then create a plan for those specific days
3. If "week" is part of the entity, *period* is set to "week" in order to address the cases where it is "week" or "weekly"
4. If "now" is part of the entity, *period* is set to "now"
5. Any other value will set *period* to "day"

After correctly setting the value for the *period*, the *time* entity will be transformed to the correct format. *Time* can be extracted in 3 different ways: a single string with a date, a list of dates, or an interval that includes the first date and the last date (from ... to ... format). The latter is useful for time periods that do not match one day or one week, such as "weekend" or "next couple of days". In case *period* is "week", the goal is to obtain a single date that belongs to that week. Bearing this in mind, the list and interval formats are converted to a single date, usually selecting the first date. On the contrary, the desired format would be a list of dates. Hence, if *period* is "day", the interval format will be transformed to a list of the dates that belong to it. Similarly, if "month" is the *period*, the time is usually extracted as the date of the first day of the month. Given this, it is necessary to create a list of the dates of all the days of that month. This is the reason behind the extended time for monthly queries. After transforming these two entities, the action checks whether the user specified any meal or not. It would be extracted as an entity and saved in the *meal_division* slot.

When all information is gathered, the request is sent to the API and according to the response status code, the chatbot sends different success or failure messages. In case the plan was successfully created, the chatbot triggers the response option that asks whether the user wants to see the created plan or not. This action is illustrated in Figure B.1.5.

5.4.1.4 Show Plan

The function to show a plan to the user, according to the different query parameters, is probably the most important and most complex group of actions coded in the agent (corresponds to use case UC.3). The high level of complexity originates in the increased variability of possible queries using all of the elements from the meal plan (day, meal and meal part). This action is divided in two main parts, where the first part is composed by one function that transforms the input information and triggers the following action from the second group. This group gathers 3 different actions that are responsible for showing the plan to the user, using buttons, for better organisation.

The first action performs an identical processing to the *time* and *period* entities, if the previous action was not the creation of the plan, in which case, it only retrieves the already correct format. The first thing that the action verifies is the presence of entities in the last user input and the existence of a previous query to show the plan. If there are no entities in the user utterance and there is a previous query, the bot will simulate that query again, since it assumes the user wants the see the same plan again.

Otherwise it verifies if a plan is created for the selected dates and if not, sends a message to the user warning that there was no plan available and asking whether the user is willing to create one. If the plan is created, the action will then retrieve it from the database (including id information for all meals) and order it chronologically, since the order is not maintained in the database. Besides ordering it according to time, it also organises the meal parts according to a defined order so that they all appear in the same way. The ordered plan and corresponding ids are then stored in the slots of the bot.

According to the entities that were extracted from the input, this first action will trigger different actions. If the *period* is "week" or "day" and the date corresponds to an interval of days it will trigger an action that shows each day as a button with the plan for that day inside it. The content of the plan will be controlled by the presence or absence of meal or meal part entities in user input. Contrarily, if the goal is to see only the meals or meal parts for one specific day, it will trigger an action that shows the meals of that day. Moreover, if the *meal* slots is set for any specific meal, it will show buttons for the meal parts the compose that meal.

The buttons showed by this and other actions serve two different purposes. The first is to organise the information into sections according to the queries. It eases the comprehension process. Also, although the chatbot mechanics work mostly by text messages, it creates an extra level of navigation that can be facilitate access to further information or activate additional actions. It is an intuitive way of selecting elements in a user interface. The buttons have a payload option that codes the information of each button and sends it to the chatbot when pressed. In this case, the buttons trigger the action that shows the next level of organisation. For example, if a user is seeing the meals for the whole week, where each button corresponds to one day, a press in one day will show the user the meals for that day. This action is illustrated in Figure B.1.4.

5.4.1.5 Nutritional Information

The user may require information about the nutritional content of each meal plan element (use cases UC.6 to UC.9). Besides these, the user may also ask for nutritional information regarding ingredients, both when seeing the plan, or as an individual query without plan reference. This function is also divided in several actions that address different plan elements. Namely, the day, the meals, the meal parts and ingredients each have different nutritional content retrieval actions. These actions take into account the data provided by the user in the form of entities and request the API for the appropriate information. The selection of which actions to take place is defined by the dialogue management model. In this case, it will take into account the level of the plan that is being showed and the extracted entities. As way to illustrate this situation, if a user is seeing the meals for one day, and asks what is the nutritional content without giving any meal entities, the model prediction should be to show the content for that day. On the flip side, if any meal was present in the user utterance the model should trigger the action that shows the nutritional content for one meal.

There is also the case where a user requests information about an ingredient. In this scenario, the algorithm will first check if there is a plan being showed. If the user is seeing the plan, it checks whether the query ingredient matches any of the ingredients present in the plan through word lemma comparison and, in a positive case it retrieves the information from that

ingredient. If there are no matches, or the chatbot is not currently showing a plan, the ingredient retrieval algorithm described in Section 5.3.2.1 is activated in order to obtain the candidates. If there is more than one candidate, the action will show the average information of all ingredients.

Regarding the button based navigation, after a button press, a message asking whether the user wants to see the next level in the plan hierarchy or the nutritional information is sent. An answer to this question referring the nutritional information will trigger the respective action in order to retrieve and present the content. An example of this action is depicted in Figure B.1.7.

5.4.1.6 Expressing Preference

Preference expression is also an important feature of this chatbot (use cases UC.10 and UC.11). Whenever the NLU component identifies the intent that corresponds to preference expression, an action that verifies whether the user is referring to a recipe or an ingredient should be triggered. This action then triggers other actions that handle the rating itself. Additionally, there are two ways of expressing preference indicating the name of the ingredient or recipe, or referring the meal or meal part.

The first action of this set works in a similar way as the one described in Section 5.4.1.5. The chatbot checks whether the user is seeing a plan or not. If so, first it tries to select the plan element based on the entities (meal, meal part, day), if they were defined in the input. Otherwise, the chatbot tries to match the query food entity with an element of the plan. However, if the case is that the bot is not showing the ingredients of a meal part and instead it is showing a day or a meal, a positive match will lead to the rating of that whole recipe. On the contrary if the ingredients are being shown, a match will proceed to the rating of that ingredient. Once again, not having a match will activate the ingredient retrieval algorithm to select the candidates that will be subsequently rated.

Following the match or ingredient retrieval, the bot updates the preference rating using the sentiment magnitude extracted from the NLU module (as an entity) and processed using the algorithm described in Section 5.2. Furthermore, in the case the rate falls in the central values, the bot asks the user if he is willing to give a numeric rating to that ingredient or recipe (from 0 to 5). The rating is then further updated and in case it is a negative value, the bot sends a message suggesting that every meal part that include that ingredient or recipe may be replaced. If the user accepts that suggestion, the conversational agent proceeds to change all meal parts that check that condition.

5.4.1.7 Alternative meals

The user has the freedom to change any meal parts that they want (use case UC.4 and UC.5). For that purpose, the same matching mechanism is used as in the previous actions. After getting the correct meal part, along with its id number, a request is sent to the API to retrieve alternatives. These alternatives are generated according to the same heuristics that guide plan creation and are explained in Section 2.2. The alternatives are then shown to the user through buttons, so that the user can select an option. Also, the selection process may be through text, where it is expected that the user refers either the ordinal position of the alternative or its name. A matching

algorithm is then activated to select the appropriate option. The last step is the substitution of the original meal for the selected alternative.

5.4.1.8 Show Personal Information

Users may want to check what personal details the agent has saved (use case UC.13). This action is triggered after detecting the intent to show profile. If no specific detail is extracted as an entity, the chatbot will show all saved information, otherwise it will address user's request.

5.4.1.9 Change Personal Information

It is important that the agent is able to update user's personal information (use case UC.14), so that the plans are generated according to the most recent data. The action is similar to the preceding one, however this one requires a specific user detail to update. The chatbot extracts the required user detail to be updated (eg. name, birth date, weight, among others) and the new value. After that, it sends a request to the API to update a specific detail with the new value.

5.4.2 Dialogue Management Dataset

Once again, there was a lack of data that could be used to train and evaluate models. For that reason, a dataset of possible paths that the chatbot can take was created through the interactive learning methodology described in Section 4.3.3.1. Dataset creation required a solid and tuned NLU module, hence it was generated after the NLU pipeline was finalised. In the first step, without any previous knowledge from the chatbot, the stories (see Section 4.3.2.2 for data format) were created by hand. When a satisfactory quantity of data was collected the model was trained so that it would be able to make some initial predictions, while alleviating the annotation process and thus increasing collection speed. Furthermore, actions that are supposed to happen immediately after an intent is identified were mapped to trigger using the Mapping Policy, described in Section 4.3.2.3. This would also facilitate predictions and dataset creation. The actions mapped to intents are represented in Table 5.22.

Table 5.22: Intent-Response mapping. The identification of any of those intents will automatically trigger the respective action as a response from the conversational agent

Intent	**Mapped Action**
create_plan	Action Create plan
show_plan	1st function of Show Plan action
show_alternative_meal	Action Show alternative meal Function that identifies the correct meal and shows alternatives
greet	Greet response
delete_plan	Action Delete plan

The next step was the population of the dataset with the help from two volunteers. A description of the use cases and the actions that encode them was given to the volunteers so that they could understand the inherent mechanics. Also, the feedback of the volunteers was taken into account to further improve the interactions between actions and agent response in a *Wizard of Oz* experiment [190] fashion.

The dataset used in to trained the model employed in the user experience testing consisted of 87 stories that correspond to a variety of paths that the conversation may take. This dataset may be further expanded using the data from testing sessions or future users.

5.4.3 Dialogue Management Experimental Evaluation

Three different approaches were tested regarding the prediction algorithm: a rule-based approach to obtain a baseline score, a Recurrent Neural Network (LSTM) based approach, and a Transformer based approach. These approaches correspond to the Memoization Policy, to the Keras Policy and to the TED Policy, described in Section 4.3.2.3. The combinations tested are shown in Table 5.23 and correspond to: DM.1 – Rule-based – Memoization Policy, DM.2 – Recurrent Neural Network – Keras Policy and DM.3 – Transformer – TED Policy. The first pipeline is expected to produce the worst results and it corresponds to the baseline score.

Table 5.23: Dialogue Management pipelines tried. The pipelines change in the used Policies for action prediction

Approach	Id	Policies
Baseline – Rule-Based	DM.1	Form Policy, Fallback Policy, Mapping Policy and Memoization Policy
Recurrent Neural Network	DM.2	Form Policy, Fallback Policy, Mapping Policy and Keras Policy
Transformer	DM.3	Form Policy, Fallback Policy, Mapping Policy and TED Policy

A similar methodology to the one used in NLU validation was taken: the models were tested three times for two different train/test ratios: 80/20 and 70/30, to assess the robustness against different training set sizes. The results are represented in Table 5.24.

Table 5.24: Dialogue Management approaches results

Model	Test percentage	F1	Accuracy
DM.1	20	0.741 ± 0.018	0.586 ± 0.027
	30	0.698 ± 0.032	0.536 ± 0.039
DM.2	20	0.859 ± 0.016	0.754 ± 0.025
	30	0.823 ± 0.029	0.701 ± 0.042
DM.3	20	0.899 ± 0.013	0.815 ± 0.025
	30	0.875 ± 0.023	0.778 ± 0.037

A correct story corresponds to predicting every single action/response correctly for the whole story. As it is possible to observe, the Memoization Policy has the worst performance of the three configurations, owing to the fact that it does not generalise well for unseen data. When comparing TED Policy with Keras Policy, it is clear that the former has a better performance demonstrating the superiority of transformer architectures. The attention module allows the model to select what is effectively important without needing a huge dataset to learn what to ignore, which is the case of the Recurrent architecture of Keras Policy. The model selected for the chatbot was the TED Policy. The inclusion of more training data is also crucial to improve performance and reach high level of quality in the agent. Ideally, the training data would contain every possible path that the agent can take.

5.5 Summary

This chapter explored the experimental evaluations for the tasks that were required to build the chatbot.

Regarding the NLU component, namely for the intent classification and entity recognition tasks, a group of experiments were performed in order to assess the best combination between features and classification models. The features tried ranged from more traditional sparse features to more recent contextual embeddings generated by Language Models. The approach that provided the best results for both tasks was a pipeline that employed both pre-trained GloVe

embeddings and Regex Features from lookup tables as well as a transformer-based classification model for joint intent classification and entity recognition. This modality of joint training has shown to improve results for both tasks.

Furthermore, extra components were added to the NLU pipeline that do not directly influence intent classification nor entity recognition, in order to enable other functions that the chatbot has to perform, namely a spellchecker, a sentiment analyser, a pre-trained entity extractor and parser and a synonym mapper.

This chapter also described the experimental evaluation that lead to the decision of including VADER sentiment analyser into the NLU components pipeline. This component is used to create a quantitative rating for a given ingredient or recipe based on the sentiment intensity and polarity mined from user input. This module plays a fundamental role in the preference expression use case, where the user should be able to indicate an inclination towards any element in the plan using natural language. The developed algorithm used the metrics extracted from VADER with some alterations inserted to properly adapt to the Lifana recommendation system.

Additionally, the food matching algorithm that is responsible for retrieving all correct ingredients from Lifana food database according to a given query entity, was depicted in this chapter. This algorithm is based on the premise that word embeddings are able to capture contextual relations between words. However, conventional word embeddings such as GloVe or Word2Vec do not fully encode semantic relations between food-related terms. In order to solve this issue, the retrofitting technique was applied in an attempt to further incorporate the semantic relations that are present in knowledge bases, more precisely **LanguaL**, into the word vectors. That being said, a matching algorithm that uses several similarity metrics, including cosine similarity between embeddings was developed. This technique's parameters can be further optimised with the help of a proper dataset.

The developmental approach to the actions that map the use cases described in Section 4.4 was also explained in this chapter, including the several heuristics and rules, as well as the required entities. The chapter ended with the description of the experimental evaluation that allowed to select a prediction pipeline for dialogue management, based on Rasa Policies. The transformer architecture showed the best performance and was selected to train the model used in the agent that was subject to user experience evaluation. In fact the next chapter explores the user experience testing that was performed for a complete and in-depth validation of the chatbot and the work done.

Chapter 6

User Experience Evaluation

A full evaluation of an interactive system requires also human judgement, besides performance metrics. This second stage of validation is very important to further refine the system and its capabilities to match user expectations. In order to do so, user experience must be analysed and studied. This chapter explores the procedure used in the assessment of human judgement regarding a finalised version of the chatbot.

6.1 Experimental Protocol – Usability Test

With a view to properly study the user experience dimension of the chatbot, a usability test was designed. Also, this type of studies requires the presence of volunteers to interact with the agent. The following sections will detail the planned test and the steps that allowed testers to use the chatbot.

Based on the work of Holmes et al. [191] that applied conventional usability and user experience metrics to evaluate the user interaction with a health chatbot, a testing pipeline was developed. The mentioned authors used two well established questionnaires: User Experience Questionnaire (UEQ) [155] and the System Usability Scale (SUS) [153], that assess both the user experience (from several perspectives further explored in the next sections) and the usability, respectively. Additionally, the authors created a chatbot oriented questionnaire that followed the scheme of the SUS, so that it could be compared against it. As mentioned in Section 3.3, there are benchmark values for these tests, which constitutes an advantage when making a comparative analysis.

That being said, the full test is composed by three parts. Owing to the agent's goal oriented nature, the first part of the test requires the volunteer to complete a set of 5 tasks that match the chatbot's most common use cases. Moreover, immediately before and after each task, the tester should answer a Single Ease Question that serves the purpose of collecting the perceived difficulty of the task (before), as well as its real difficulty (after), on a scale from 1 to 7, where 1 is very hard and 7 is very easy.

Following the completion of the tasks, the user is then instructed to answer three questionnaires: SUS, UEQ and the chatbot focused usability questionnaire. The third and last step is the validation of the ingredient retrieval algorithm described in Section 5.3.2.1. The questionnaires were sent using a Google Form, which allows easy analysis, using the spreadsheet tool. The chatbot was deployed on a Google Cloud Virtual Machine using Rasa-X. This creates a conversation interface that can be send to the testers by sharing a URL. The steps of the protocol are the following:

1. The volunteer signed the consent form
2. Access to the chatbot and form was given to the volunteer
3. Test instructions and details were explained to the volunteer (see Section B.2.1)
4. The volunteer completed the demographic questionnaire (pre-testing)
5. Each task was read to the volunteer which confirmed their understanding
6. The volunteer answered the pre-task Single Ease Question
7. The volunteer performed the task under the supervision of the coordinator
8. The volunteer answered the post-task Single Ease Question
9. After all tasks are completed, the volunteer completed the usability surveys described above
10. Posterior to the surveys, the ingredient retrieval questions were answered
11. At the end the volunteer gave any extra feedback on the agent, that may be used to further improve it as future work

Also, the task's time to completion was recorded so that further comparisons may be made. The protocol was conducted remotely due to social distancing constraints.

6.1.1 Test Tasks

The first part of the test is composed by the following 5 tasks:

1. Chatbot's welcoming process

   The task consists in creating a user profile using the chatbot. The agent requests several personal details sequentially to fill the slots and create the profile.

2. Generate a meal plan and see the ingredients

   The volunteer must command the bot create a meal plan for monday lunch and check the ingredients for the main dish

3. Check the plan and express preference

   The volunteer has to ask to see the plan for monday dinner and express the preference towards one ingredient of it

4. Create a plan, check the meals and ask an alternative

   This task is complete after creating a meal for the next couple of days, checking one of the day's dinner dessert and substituting it for an alternative

5. Check the plan and ask for nutritional content

   The volunteer has to get the nutritional content for a day/meal/part/ingredient, after checking it

The tasks have increasing difficulty and are composed by sub-tasks that repeat from task to task, so that it is possible to understand the learning curve of using the chatbot.

6.1.2 System Usability Scale

As mentioned in Section 3.3, SUS is a Likert Scale that aims at evaluating interactive systems from a usability point of view. It consists of 10 opinion scoring questions where the user must refer their level of agreement towards a sentence. The scores obtained from this test can be compared to benchmark values in order to understand the overall usability of this system when compared to an established scale. More specifically, SUS scores above 68.0 are considered to be above average according to a study conducted by Bangor et al. [192]. The same study graded the scores using adjectives or letters from "Worst Imaginable" (scores 0 - 25) to "Best Imaginable" (scores > 84.1). The same authors in another study related SUS scores with acceptability and concluded that a score above 71.1 means that the system is "Acceptable" however, a score below 62.6 classifies the system as "Not Acceptable" [193]. Moreover, SUS is also related to the Net Promoter Score, which tends to measure user loyalty [154]. A score above 78.8 means the user is a "Promoter" while a score below 62.7 classifies them as "Detractor". Values that fall between those will classify as "Passive" [194].

6.1.3 User Experience Questionnaire

The UEQ [155] tries to extensively evaluate user experience. It is based on six factors resumed in Table 6.1 and assessed through a questionnaire that asks the user level of agreement with several pairs of opposite adjectives. It measures the extent to which users expectations were met by the system. Once again the scores may be compared to a benchmark [195].

Table 6.1: Factors evaluated by the User Experience Questionnaire

Factor	Description
Attractiveness	Extent to which users "like" the system
Perspicuity	Ease of learning and becoming proficient in the system
Efficiency	Effort required to complete tasks, System reaction times
Dependability	Extent of user control, System predictability/security
Stimulation	How fun/exciting is it to use the system
Novelty	Creativeness/Interest for users

6.1.4 Chatbot Usability Questionnaire

The Chatbot Usability Questionnaire (CUQ) was built by Holmes et al. [191] as a way to appropriately measure usability scores for chatbots. It takes into account several factors that are intrinsic to a conversational agent experience: Personality, Onboarding, Understanding, Answering, Navigation, Error management and Intelligence. Bearing this in mind they developed 16 questions using the same method as SUS, the questions are resumed in Table 6.2 The scores obtained in this questionnaire can be normalised to match the range provided by SUS and further compared.

Table 6.2: Chatbot Usability Questionnaire questions.

Questions
The chatbot's personality was realistic and engaging
The chatbot seemed too robotic
The chatbot was welcoming during initial setup
The chatbot seemed very unfriendly
The chatbot explained its scope and purpose well
The chatbot gave no indication as to its purpose
The chatbot was easy to navigate
It would be easy to get confused when using the chatbot
The chatbot understood me well
The chatbot failed to recognise a lot of my inputs
Chatbot responses were useful, appropriate and informative
Chatbot responses were not relevant
The chatbot coped well with any errors or mistakes
The chatbot seemed unable to handle any errors
The chatbot was very easy to use
The chatbot was very complex

6.1.5 Food Matching Validation

After the surveys, the volunteer must suggest an ingredient (one that has not yet been selected by other testers). Both the results obtained through the ingredient retrieval algorithm (based on embeddings) and the results using the search function implemented in the API (fuzzy matching based in the name) are shown to the volunteer who has to answer to three questions.

1. Whether or not the options retrieved by the embeddings algorithm are correct on a scale of 4 values from "totally incorrect" to "totally correct", including "mostly incorrect" and "mostly correct".

2. If there is any option on the search results that should also be in the embeddings results list. The answers to this question can range from "none missing" to "all missing", including "a few missing" and "many missing"

3. What is the preferred option for ingredient retrieval

This way it is possible to evaluate the developed algorithm from a user standpoint, using some approximate recall and precision metrics as well as preferred method.

6.2 Results

A total of 22 volunteers performed the test, where most of them have already used a conversational agent or a smart assistant at least once. However, the majority does use them less than once a month (see Figure 6.1). The volunteers age did not match the end-user age due to social distancing constraints.

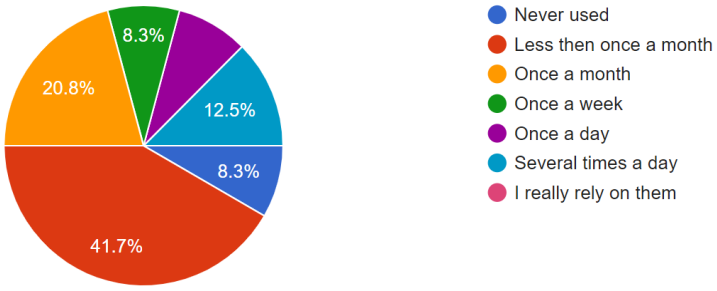


Figure 6.1: Volunteer chatbot use frequency

6.2.1 Single Ease Question

Regarding the perceived and real difficulty of the tasks, it is possible to see in Figure 6.2 that, on average, the tasks are perceived as less easy than their real easiness. By looking at Figure 6.3, that analyses the difference between real and preconceived difficulty per task, the same idea is reinforced: the tasks are easier than what volunteers initially thought.

Excluding Task 1, which is not plan-related, Task 4 presented the lowest average difference. However, this task is the one that requires the most steps and its difference value is very close to the one calculated for Task 2 which is the the first plan related task that the volunteer must perform. This shows that, even though Task 4 is the most complex, the learning curve is fast enough that volunteers feel as confident as when performing Task 2, which is the first contact.

The value for the difference also increases from task to task, showing that users can learn to work with the bot easily and become ever more comfortable. In fact, the highest difficulty difference value is recorded for Task 5 which is the one that gives more freedom to the user.

Also, another indicator of increased easiness perception is the almost constant value of preconceived difficulty along the several tasks, even though they increase in complexity (see Figure 6.3. Furthermore, looking at the absolute values of real easiness, they are close to the maximum (7) for three tasks, whereas the other two are Task 2 (the first plan related) and Task 4 (the most complex).

Regarding Task 1, the same pattern arises, the preconceived value for how easy it will be was lower than the real easiness of the task.

With a view to further support these observations, T-Student pairwise probability was calculated for pre and post scores. The probability values, presented in Table 6.3, show that these distributions are statistically different, with $p < 0.05$.

This suggests that volunteers start to feel more comfortable with the system along the course of the test session.

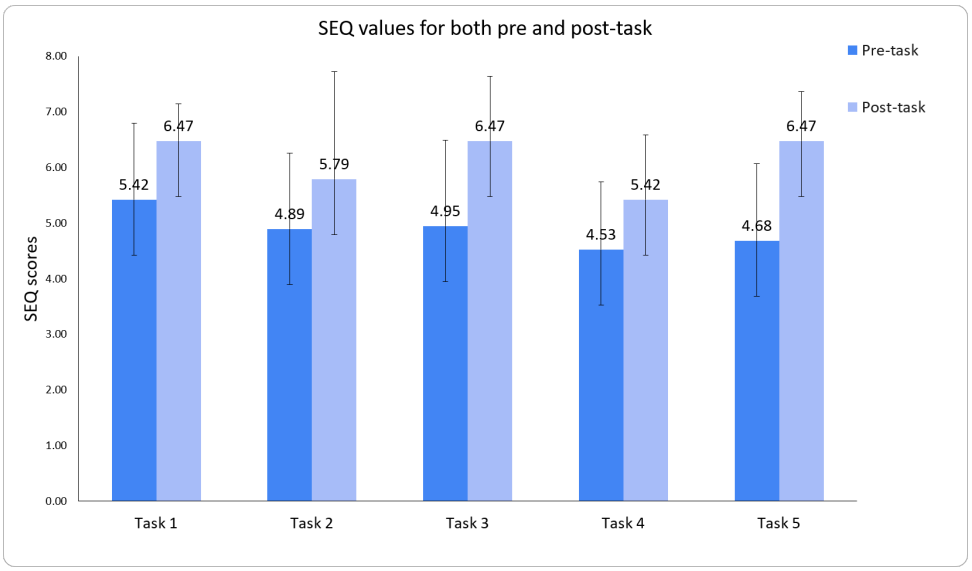


Figure 6.2: Average values for Pre-task and post-task difficulty assessment question

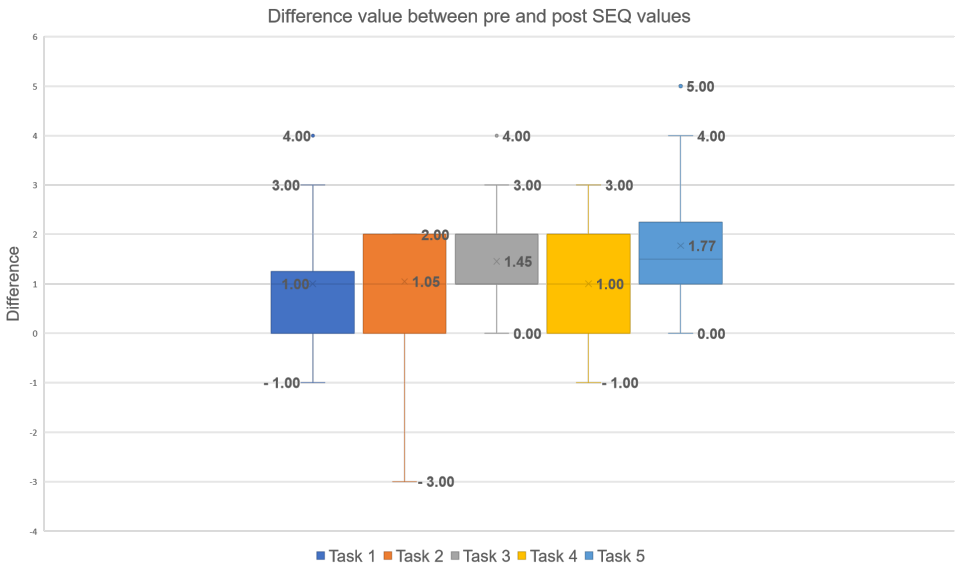


Figure 6.3: Average difference values between preconceived ease and real ease per task

Table 6.3: t-Student pairwise test probability. The values are all below 0.05, meaning that the difference is statistically significant.

	Task 1	Task 2	Task 3	Task 4	Task 5
t-Student probability	3.9×10^{-4}	5.9×10^{-4}	1.12×10^{-6}	5.6×10^{-4}	1.12×10^{-6}

6.2.2 Completion times

Figure 6.4 shows the average time to completion for each task in seconds. A similar pattern to the one observed in the difficulty assessment emerges in these results: the time to completion decreases from Task 2 to the last task excluding Task 4 which requires more steps. Moreover, the time taken by the developer is also represented in the graphic as a benchmark. It is possible to see that this value is consistently lower than the average time for the volunteers. Howbeit, the developer time falls into the error intervals for the tasks that inherently require more steps (Tasks 1 and 4). This may suggest that in Task 4 the bottleneck is not the ability or knowledge about the system, but instead the chatbot inference times.

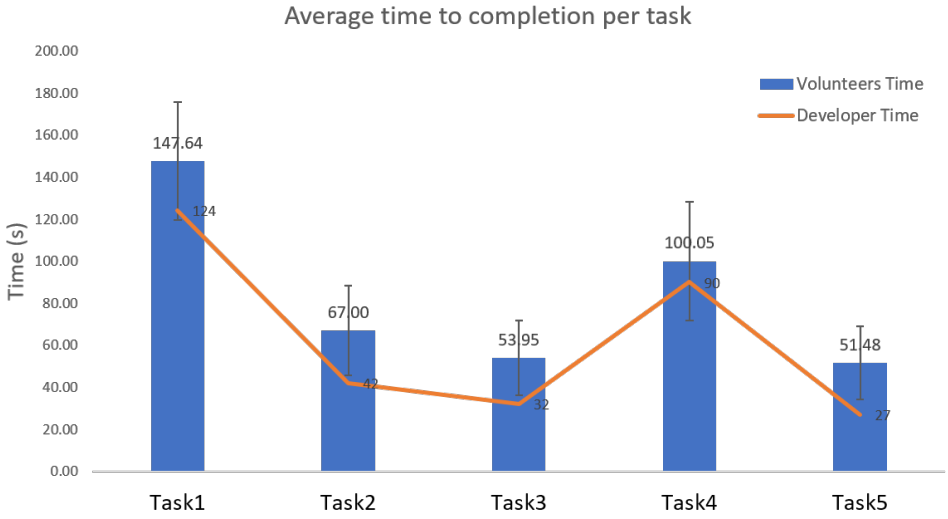


Figure 6.4: Average time to completion per task for the volunteers. The time taken by the developer is also represented by the line as a term of comparison

6.2.3 System Usability Score

The final average SUS score for the system was 84.7 ± 8.4 which positions this system as above the benchmark score, "Excellent" in the SUS adjective grading system and "Acceptable" in the acceptability scale. The lower score was 67.5 and the highest was 97.5 (see Figure 6.5). Hence this system has a high degree of usability when considering traditional SUS scores, although these benchmark scores have not included usability scores from conversational agents.

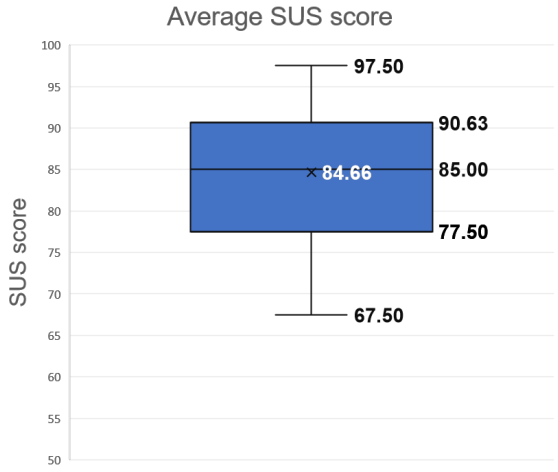


Figure 6.5: Box-plot representing the SUS scores obtained in the test. The average is 84.7, the minimum was 67.5 and the maximum was 97.5. It is possible to observe that the first quartile is above the benchmark value 68.0

6.2.3.1 User Experience Questionnaire

This questionnaire produces 6 different metrics according to the factor that it measures. The scores were normalised to a range that goes from -3 to 3 and are shown in Table 6.4. The values were also compared against the benchmark for each factor and it can be consulted in Figure 6.6.

Table 6.4: UEQ results for each factor and comparison to the Benchmark. The average calculated by averaging each volunteer mean result is also presented (scale: [-3;3])

Scale	Mean	Std. Dev	Comparison to benchmark	Interpretation
Attractiveness	1.98	0.77	Excellent	In the range of the 10% best results
Perspicuity	2.31	0.52	Excellent	In the range of the 10% best results
Efficiency	1.82	0.71	Good	In the range of the 10% best results
Dependability	1.76	0.76	Excellent	In the range of the 10% best results
Stimulation	1.84	0.66	Excellent	In the range of the 10% best results
Novelty	1.90	0.68	Excellent	In the range of the 10% best results
Average	1.93	0.58	-	

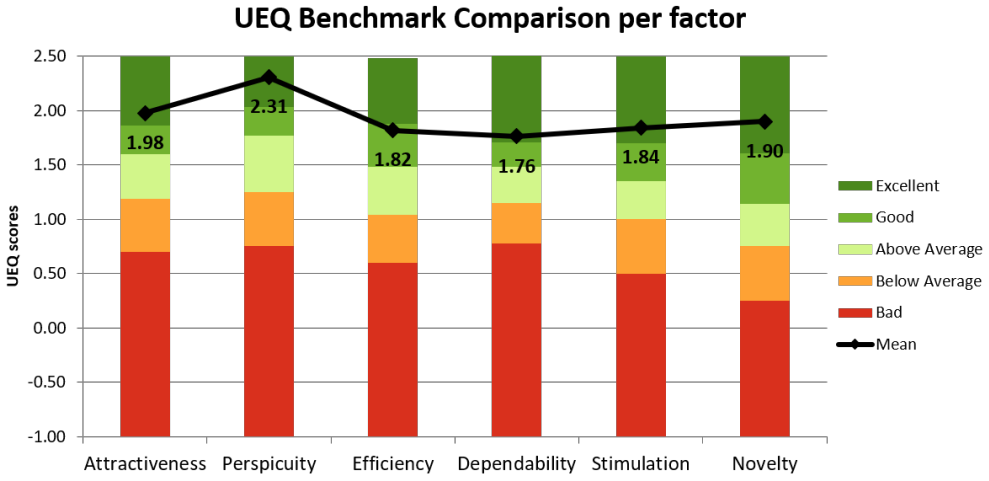


Figure 6.6: Benchmark comparison for each factor evaluated by UEQ

As shown in the previous table, the chatbot was classified with high scores in all factors when compared to the benchmarks. The factors that received the worst results were *Efficiency* and *Dependability*, even though the latter is still considered as Excellent.

Regarding *Efficiency*, the low score may be due to the increased time that some requests to the API take to process, which is the major bottleneck in terms of task duration. This is perceived as a chatbot's drawback. Moreover, the user interface used for deployment (Rasa X) had some flaws, which may also affect the perception of the chatbot's efficiency.

Dependability analysed indicators such as predictability, given support, security and expectations. Security is not possible to properly evaluate by the test that was designed, which led to a high number of mid range results. Moreover, predictability and support are the indicators with lowest score among the four. In fact, these indicators have to be improved, namely in the failing scenarios, which ideally would be none. However, in case that happens, either due to a problem of a chatbot or to a problem concerning the API, clear and adequate feedback should be given to the user. This can be achieved by optimising the models and the confidence thresholds and creating case-specific feedback responses for each error. This is, in fact, the part of the conversational agent that has the largest room for improvement.

On the other hand, the best value is achieved for *Perspicuity* which takes into account system understandability, easiness, complexity and clarity. These results reinforces the observation made in Section 6.2.1 which state that the system is indeed very easy to learn, with a fast learning curve. This is highly desirable for the elderly population to adopt and engage this technology.

In order to further compare this test with the other questionnaires, the average of the 6 factors was calculated for each volunteer and will be used in the next Section.

6.2.4 Chatbot Usability Questionnaire

The results obtained for this questionnaire were normalised to the same interval used by SUS (0-100), so that it is possible to compare them and to the benchmarks.

The average score for CUQ was 82.9 ± 8.6. Once again it is well above the benchmark value of 68.0 and positions the chatbot in the same categories as the SUS score.

The maximum CUQ value is higher than the maximum SUS value and the minimum CUQ value is lower than the minimum SUS value. However, the distribution of the CUQ values is more compact than that of the SUS because the interquartile range is smaller (see Figures 6.5 and 6.7). Also, the results may show a slight positive skew since the median is lower than the mean.

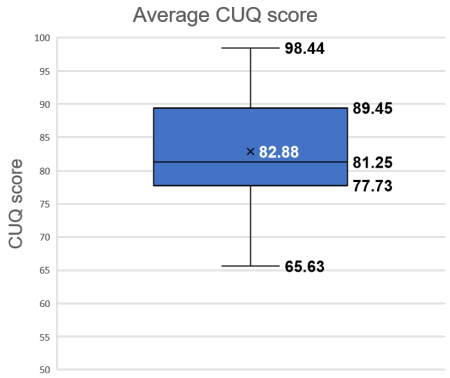


Figure 6.7: Box-plot representing the CUQ scores obtained in the test. The average is 82.9, the minimum was 65.6 and the maximum was 98.44.

6.2.5 Questionnaire Comparison

An important analysis to be made is whether the different questionnaires evaluate the chatbot in the same way or not. Figures 6.8, 6.9 and 6.10 show the several questionnaire scores plotted against each other. In general terms, the relationships between the several questionnaire scores show a linear propensity.

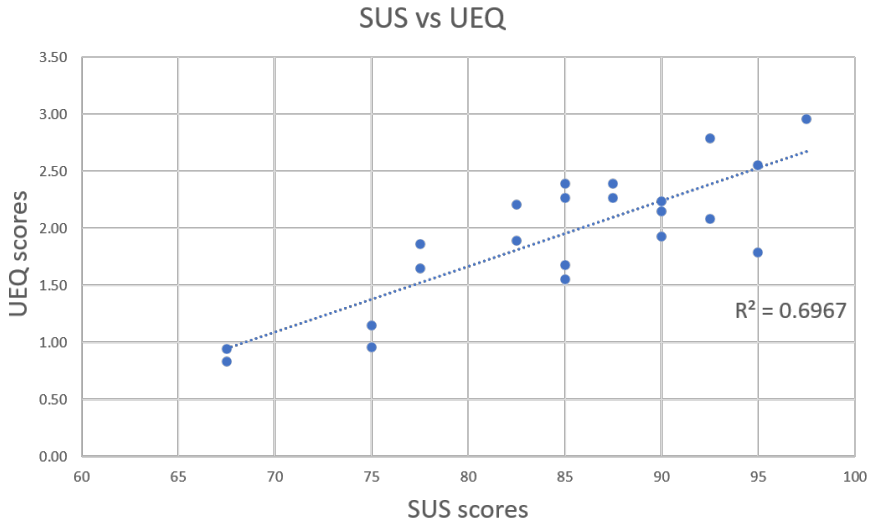


Figure 6.8: SUS vs UEQ scores per volunteer plot

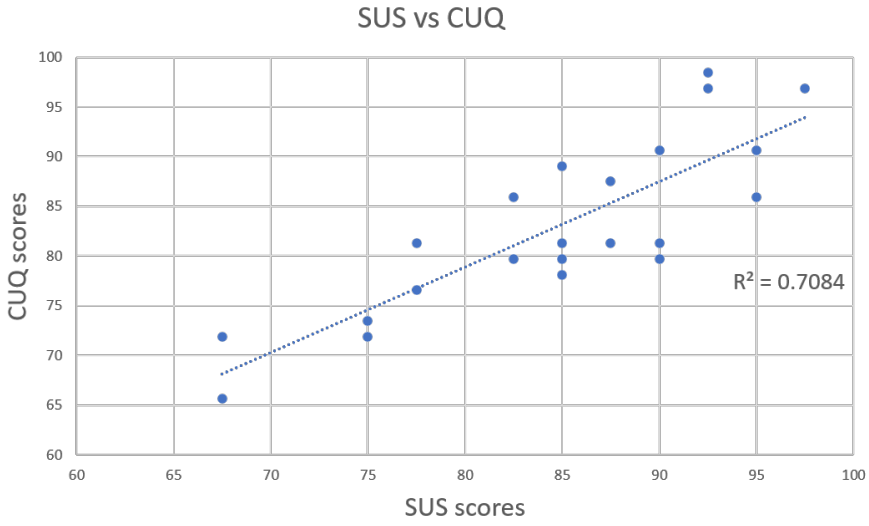


Figure 6.9: SUS vs CUQ scores per volunteer plot

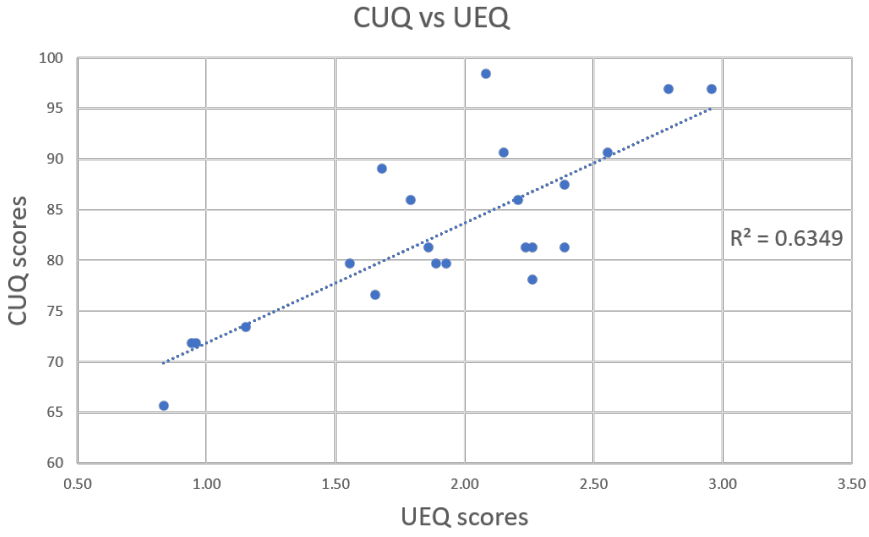


Figure 6.10: UEQ vs CUQ scores per volunteer plot

In order to confirm this trend, the Person correlation was calculated between each pair of questionnaire scores (see Table 6.5). The UEQ values were normalised for the 0-100 scale. The values are all close to 1 which means that the relation can be approximated by an increasing linear function. The highest value was obtained for the SUS and CUQ. This fact might be due to the CUQ being inspired in the SUS.

Table 6.5: Pairwise Pearson correlation values for the several questionnaires used

	SUS & UEQ	**SUS & CUQ**	**UEQ & CUQ**
Pearson correlation	0.835	0.842	0.797

CUQ score is lower than SUS, however, the difference between them is not statistically significant ($p < 0.05$). Actually, the t-Student probability was calculated for each pair and the results may be consulted in Table 6.6. The difference between SUS and UEQ scores is statistically significant ($p < 0.05$), whereas the difference between UEQ and CUQ is not statistically significant, even though there is a higher linear correlation between SUS and UEQ than CUQ and UEQ. This may suggest that CUQ and UEQ are measuring similar aspects of the chatbot. Also, considering that the R^2 corresponds to the overlap between what the questionnaires are measuring, it is possible to say that approximately 63% of the variance in CUQ score can be explained by UEQ. Hence, 37% of this value might be related to other factors that are more closely related to conversational agents that is not being captured by UEQ.

Table 6.6: t-Student probabilities for pairwise combination of each questionnaire

	SUS & UEQ	**SUS & CUQ**	**UEQ & CUQ**
t-Student probability	0.045	0.098	0.610

6.2.6 Food matching

Due to the lack of an adequate dataset, the validation of the food matching algorithm was performed through user questionnaire. The questions tried to assess the recall and precision of the developed algorithm. Figure 6.11 shows the answer distribution for the question that addressed the correctness of the shown results by the embeddings-based search algorithm. The results presented can be mapped to the precision of the system since it measures the how many of the positive answers are in fact true. The results show that the developed algorithm has a high level of precision. The largest part of the query outputs are totally correct meaning that the ingredients that it shows are in fact related to the query term.

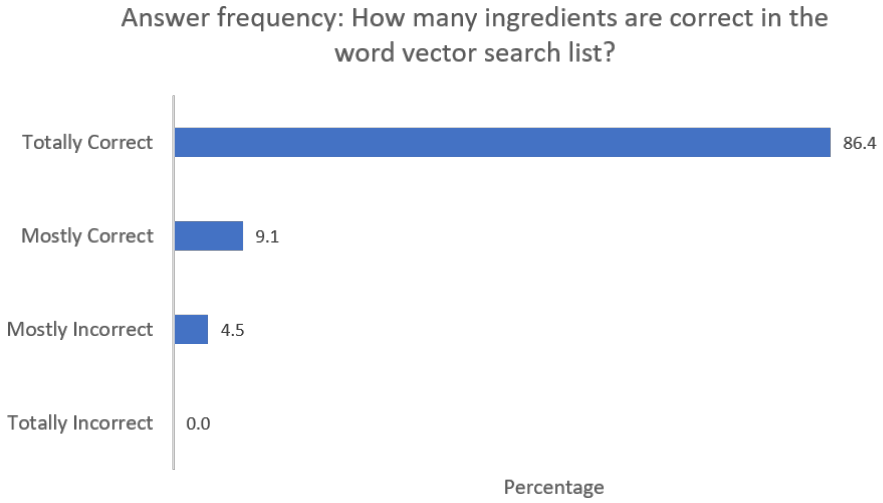


Figure 6.11: Response frequency for food matching precision assessment

Recall is also another important metric that should be taken into account. Figure 6.12 shows the answer frequency regarding the comparison made between the already implemented in the API word-based search and the developed algorithm. The goal was to identify items that were correctly present in the former and missing in the latter. This would not calculate the true recall of the model, since this would require a list of all correct items per query. Nonetheless it is a good comparison to detect missing items. The results show that the largest part of the answers were positive, meaning that the recall is also high. However, it is possible to affirm that the model shows increased precision when compared to recall.

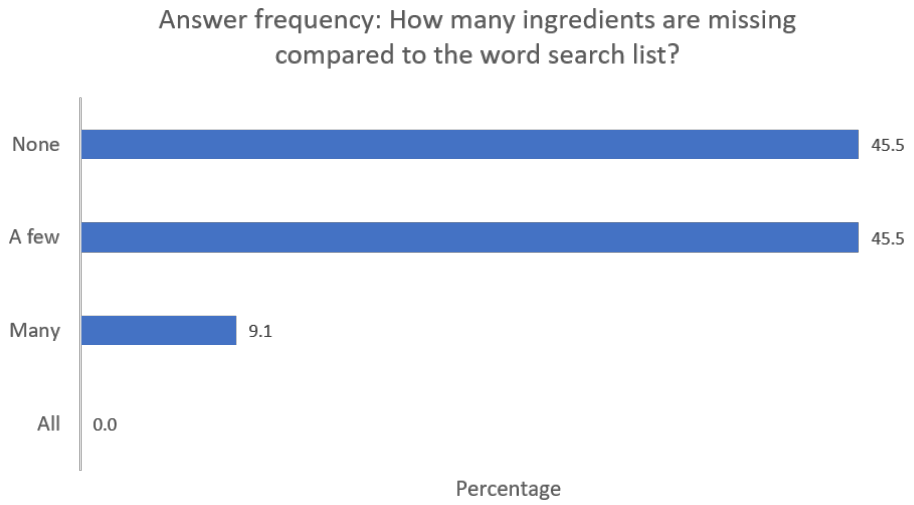


Figure 6.12: Response frequency for food matching recall assessment

Figure 6.13 shows the answers to the question that judged which search algorithm was preferred by users. It shows that, even though the recall is lower than precision and the algorithm do not always gathers all ingredient samples from the dataset, users prefer to have the rating applied only to correct ingredients. Imagining a 0 valued rating (ingredient removal from the plans), this suggests that users would rather deal with incorrectly present ingredients than with incorrectly removed ingredients.

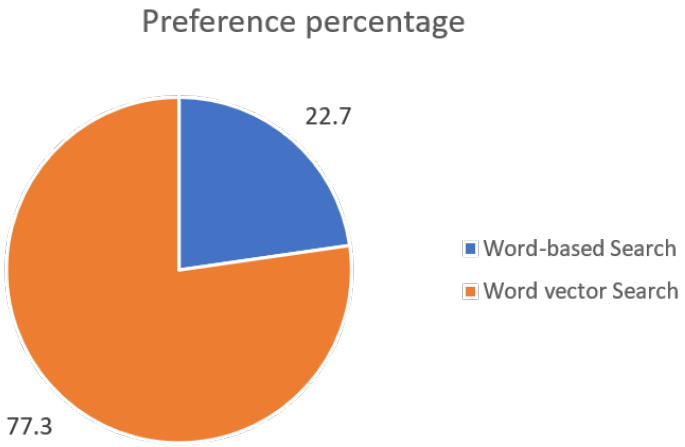


Figure 6.13: Response frequency for food matching model preference

6.3 Summary

This chapter explores the usability and user experience testing that is crucial for a complete chatbot evaluation. It starts by describing the used protocol which is divided in two parts: the task testing and the questionnaires. The statistical analysis of the obtained results is presented is also presented and discussed in this chapter.

Overall, the conversational agent developed in this work showed good results when compared to benchmark values for each test, although subsequent improvements can be made. Also, a joint analysis to questionnaires is performed, in order to assess their appropriateness regarding chatbot evaluation. Furthermore, the food retrieval algorithm was also evaluated resorting to user opinion. The results show that even though there is still room for optimisation, the algorithm provides good results and is the preferred approach when compared to a word matching search.

The analysis conducted in this chapter provides valuable insights that may guide further development of this chatbot, as well validate some decisions that were taken along the development phase.

Chapter 7

Conclusion and Future Work

Malnutrition is a disorder that affects billions of people worldwide and its main cause is the lack of an appropriate diet. The high prevalence and incidence of malnutrition related diseases represent an enormous social and economic burden on the world. That being said, it makes sense to study and develop dietary plans and recommendations that help the population to ingest a healthier selection of foods and correctly prepared meals. However, the creation of a meal plan or definition of dietary requirements depends on the individual user and its characteristics. Furthermore, it requires extensive human resources and labour to analyse each case and provide a suitable dietary plan. Given these limitations, nutrition companions have been emerging as a tool that permits user to easily access relevant nutritional information and meal plans. In fact, these companions usually correspond to mobile application that is simply installed in the smartphone which is a widely available asset. Nevertheless, the interaction between these systems and the user happens through a screen and touch-based inputs. This form of interaction can be confusing and often discouraging for the elderly population.

The necessity of a more immersive form of interaction becomes evident in this context, in order to stimulate an extended and increased usage rate of these nutrition recommendation systems. *Franhofer's* Lifana recommendation system is an example of such a technology that could be improved by the implementation of a revamped form of interaction. That is the underlying motivation for the creation of a conversational agent to be integrated in the system. Thus, this document comprehends the basics for a Master's Thesis that addresses the aforementioned needs.

The underlying research to this work covers the techniques and algorithms that are often employed in the construction of conversational agents. The natural language understanding methods are the core of the chatbot, while speech recognition and speech synthesis also play important roles in the quality of the interaction of the agent. These two components are not part of this work's scope, even though their implementation would fit easily in the pipeline of the chatbot. As a matter of fact, speak detection would ideally solve the issue with misspellings since its output is always correctly spelled. Moreover, the agent being developed for fully text-based conversations allows for seamless integration with these modules.

That being said, the identification of the user's underlying intention in a utterance and

the prediction of the next chatbot's response are the key functions for the proper functioning of a task-oriented chatbot. Both these tasks were performed using Rasa framework which included a set of state-of-the-art tools that facilitated the development of the chatbot. In order to overcome the initial lack of a proper dataset, two correctly annotated datasets were collected: a crowd-sourced NLU focused set of utterances which were then adjusted and annotated and also a dataset of human-chatbot turn-based interactions, collected with the help two volunteers through Rasa Interactive Learning option. The goal of these sets was to bootstrap the creation of the chatbot and it is recommended that they are further expanded with real user data. This will lead to a boost in the performance of the chatbot, with less fail cases or better failure scenarios. The option of using sentence structures with varying slots for data augmentation was considered, although the utilisation of real data was regarded as a better option. The absence of data guided many options taken during the work and hindered the validation of some components added. Nonetheless, with the collection of more data, this work set the stage and support for an ever improving virtual agent.

The NLU component developed during this work had the tasks of classifying intent and extracting entities from a user input. Several architectures were tried, including one transformer-based architecture with an attention head which is the current state-of-the-art for NLP problems. Also, traditional machine learning models were used, including an SVM, which generated very better results for intent classification than the transformer model and a CRF for entity recognition. Moreover, different sets of features representing increasing levels of complexity were also experimented. These features range from traditional token-level sparse features to both pre-trained and contextual word embeddings generated from cutting-edge language models. It would be expected that the model using the more recent and complex contextual embeddings would outperform the other methods, however, owing to the closed-domain of this work, the best results for both tasks were obtained using static pre-trained word embeddings. Both intent classification and entity extraction were handled in a multitask fashion where the classification model used the combined loss of both problems to guide training. This method was achieved using the DIET transformer architecture developed by Rasa and displayed the best results overall.

Once the NLU pipeline was curated and tested, it was possible to focus the efforts on the Dialogue Management part. Several architectures were tested and the transformer-based architecture was selected due to its higher performance scores. The testing concerning this part were not as extensive as that of the NLU since the collected dataset was much smaller. The focus was to build a model and a dataset that worked together well enough for a proper functioning of the chatbot. This dataset is easily expanded by storing tester conversations that may be subsequently annotated. That being said, the chatbot performance will benefit from this collection process.

With a view to complement the chatbot and improve the quality of the recommendations, food preferences had to be modelled according to the user's input. For this reason, two extra components were developed: the preference modelling and the food matching algorithms. The former was accomplished using a sentiment analysing tool that fit in the NLU pipeline. This required an in-depth analysis of two sentiment analysers. The results were explored to create an appropriate rating mechanism based on the data collected in the NLU form. This component ex-

tracted the sentiment intensity present in the user utterance when referring a preference towards an ingredient or recipe. This magnitude value was then converted to the correct scale. Moreover, the rating algorithm used by the API had be to taken into account in order to adapt the sentiment-based rating accordingly. The values have different weights and hence diverse effects on meal planning. This component can be further optimised with the collection of more preference data. In fact, with enough data, a classification model may be trained from end-to-end which will better capture the information present in the dataset and adapt itself to the API scoring mechanism. To a proper functioning of this preference model, the calculated rating should be applied to the correct set of ingredients mentioned by the user. In fact, the matching process between the query entity extracted from the text and the several ingredients in the database is not trivial due to both the presence of various instances of each ingredient and to the inherent semantic relations present in this lexical set. Bearing this mind, a matching algorithm based on embeddings was developed in an attempt to capture the semantic relations between food terms. These relations are encoded as a knowledge graph in the *LanguaL* food ontology, which was used as the standard for ingredient classification. This semantic information encoded through *LanguaL* was incorporated into a set of pre-trained word embeddings using retrofitting. The chosen set was the ConceptNet Numberbatch, since it was already retrofitted using a general purpose knowledge graph and hence its baseline results were better than using GloVe embeddings. The set of word vector obtained after retrofitting with *LanguaL* is used as a base for the matching algorithm. It works by applying sequential filtering steps to the candidates, based on several metrics of similarity. The final goal is to obtain a list of ingredients that match the query entity, either if it is a group of ingredients or if it is an individual one. The similarity thresholds defined during the development of this algorithm were based on visualisation and can be optimised with a dataset created for this purpose. Moreover, the generated set of embeddings now include the semantic relations present in food items. Hence, this information can be further explored for different applications such as ingredient substitution or recipe creation by leveraging the relations between different ingredients in the database. Also, the combination of these embeddings with sets like the *food2vec* may pose as a good resource for automatic recipe suggestion or creation based on personal preferences, increasing the number of recipes available in the database. The developed algorithm was validated by user opinion. The conclusion taken from the analysis of those results is that users prefer precision over recall when dealing with rating ingredients. This means that overall, the embeddings-based algorithm was preferred over the word search. The algorithm usually fails in retrieving all samples when they are part of a complex ingredient that includes several others. The classification of those samples is usually based on the main ingredient. Another way of improving the performance is by altering these complex ingredients (that sometimes are complete dishes) by splitting them into their components. This suggestion would also facilitate recipe creation and ingredient substitution since the recipes become more modular.

In order to perform a complete assessment to the developed chatbot, user experience should be taken into account. Therefore, a usability test was designed based on singular tasks. The selected tasks are inspired by the most common use cases of the chatbot and try to explore every action of it. Also, the tasks are design to increase in complexity and in the degree of freedom given to the user. Subsequent to the tasks, the volunteer is requested to answer to three questionnaires that aim at assessing the overall system quality. The metrics extracted from the

questionnaires and its comparison with the benchmarks allowed to conclude that the chatbot is well built and its functions are well integrated, from a usability point of view.

A crucial aspect that may be improved is the failing mechanics of the chatbot. It should cope better with errors and present detailed feedback of the problems that happen both in the conversational part and in the requests to the API. This is one of the factors that must become more robust for a possible commercialised application.

The modular development adopted in this work enables the conversion from the English language to any other language as long as there are linguistic resources available, such as sets of pre-trained word embeddings and an appropriate dataset.

Altogether, the realisation of this work was fundamental for a complete understanding of the requirements and techniques that are involved in the construction of a conversational agent that should be capable of enhancing the interaction and engagement between a user and a recommendation system. The framework used, Rasa, enabled quick and efficient development and integration between all the components. The final product can be implemented in several chat applications such as Facebook Messenger, Telegram or even Slack, without the need to download an extra application, or as a built-in feature in other devices such as smart cooking appliances (with the addition of voice controls).

Bibliography

[1] *Database - Eurostat*. [Online]. Available: https://ec.europa.eu/eurostat/web/population-demography-migration-projections/population-projections-/database.

[2] World Health Organization, "World report on ageing and health", World Health Organization, Geneva, Switzerland, Tech. Rep., 2015.

[3] D. Baez-Franceschi and J. E. Morley, "Physiopathology of the catabolism associated with malnutrition in the elderly", in *Malnutrition in the Elderly*, Steinkopff, 1999, pp. 19–29. DOI: 10.1007/978-3-642-47073-8{_}4.

[4] S. S. Schiffman, "Taste and smell losses in normal aging and disease.", *JAMA*, vol. 278, no. 16, pp. 1357–62, ISSN: 0098-7484. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/9343468.

[5] N. Kshetrimayum, C. V. K. Reddy, S. Siddhana, M. Manjunath, S. Rudraswamy, and S. Sulavai, "Oral health-related quality of life and nutritional status of institutionalized elderly population aged 60 years and above in Mysore City, India", *Gerodontology*, vol. 30, no. 2, pp. 119–125, Jun. 2013, ISSN: 07340664. DOI: 10.1111/j.1741-2358.2012.00651.x. [Online]. Available: http://doi.wiley.com/10.1111/j.1741-2358.2012.00651.x.

[6] R. O. Hansson, W. H. Jones, B. N. Carpenter, and J. H. Remondet, "Loneliness and Adjustment to Old Age", *The International Journal of Aging and Human Development*, vol. 24, no. 1, pp. 41–53, Jan. 1987, ISSN: 0091-4150. DOI: 10.2190/82XB-5L9T-JWKE-U6T8. [Online]. Available: http://journals.sagepub.com/doi/10.2190/82XB-5L9T-JWKE-U6T8.

[7] M. P. Thompson and L. K. Morris, "Unexplained Weight Loss in the Ambulatory Elderly", *Journal of the American Geriatrics Society*, vol. 39, no. 5, pp. 497–500, 1991, ISSN: 15325415. DOI: 10.1111/j.1532-5415.1991.tb02496.x.

[8] K. Shifler Bowers, E. Francis, and J. L. Kraschnewski, *The dual burden of malnutrition in the United States and the role of non-profit organizations*, Dec. 2018. DOI: 10.1016/j.pmedr.2018.10.002.

[9] W. J. Evans and D. Cyr-Campbell, "Nutrition, exercise, and healthy aging", *Journal of the American Dietetic Association*, vol. 97, no. 6, pp. 632–638, 1997, ISSN: 00028223. DOI: 10.1016/S0002-8223(97)00160-0.

[10] C. Evans, "Malnutrition in the Elderly: A Multifactorial Failure to Thrive", *The Permanente Journal*, vol. 9, no. 3, Jul. 2005, ISSN: 15525767. DOI: 10.7812/tpp/05-056.

[11] C. C.-H. Chen, L. S. Schilling, and C. H. Lyder, "A concept analysis of malnutrition in the elderly", *Journal of Advanced Nursing*, vol. 36, no. 1, pp. 131–142, Oct. 2001, ISSN: 0309-2402. DOI: 10.1046/j.1365-2648.2001.01950.x. [Online]. Available: http://doi.wiley.com/10.1046/j.1365-2648.2001.01950.x.

[12] Development Initiatives, "Global Nutrition Report Shining a light to spur action on nutrition 2018", Development Initiatives, Bristol, UK, Tech. Rep., 2018.

[13] *Malnutrition*. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/malnutrition.

[14] A. G. Rojer, H. M. Kruizenga, M. C. Trappenburg, E. M. Reijnierse, S. Sipilä, M. V. Narici, J. Y. Hogrel, G. Butler-Browne, J. S. McPhee, M. Pääsuke, C. G. Meskers, A. B. Maier, and M. A. de van der Schueren, "The prevalence of malnutrition according to the new ESPEN definition in four diverse populations", *Clinical Nutrition*, vol. 35, no. 3, pp. 758–762, Jun. 2016, ISSN: 15321983. DOI: 10.1016/j.clnu.2015.06.005.

[15] J. I. van Zwienen-Pot, M. Visser, M. Kuijpers, M. F. Grimmerink, and H. M. Kruizenga, "Undernutrition in nursing home rehabilitation patients", *Clinical Nutrition*, vol. 36, no. 3, pp. 755–759, Jun. 2017, ISSN: 15321983. DOI: 10.1016/j.clnu.2016.06.003.

[16] M. Visser, D. Volkert, C. Corish, C. Geisler, L. C. de Groot, A. J. Cruz-Jentoft, C. Lohrmann, E. M. O'Connor, K. Schindler, and M. A. de van der Schueren, "Tackling the increasing problem of malnutrition in older persons: The Malnutrition in the Elderly (MaNuEL) Knowledge Hub", *Nutrition Bulletin*, vol. 42, no. 2, pp. 178–186, Jun. 2017, ISSN: 14719827. DOI: 10.1111/nbu.12268. [Online]. Available: http://doi.wiley.com/10.1111/nbu.12268.

[17] H. Kruizenga, S. van Keeken, P. Weijs, L. Bastiaanse, S. Beijer, G. Huisman-de Waal, H. Jager-Wittenaar, C. Jonkers-Schuitema, M. Klos, W. Remijnse-Meester, B. Witteman, and A. Thijs, "Undernutrition screening survey in 564,063 patients: patients with a positive undernutrition screening score stay in hospital 1.4 d longer", *The American Journal of Clinical Nutrition*, vol. 103, no. 4, pp. 1026–1032, Apr. 2016, ISSN: 0002-9165. DOI: 10.3945/ajcn.115.126615. [Online]. Available: https://academic.oup.com/ajcn/article/103/4/1026/4662899.

[18] P. Abizanda, A. Sinclair, N. Barcons, L. Lizán, and L. Rodríguez-Mañas, *Costs of Malnutrition in Institutionalized and Community-Dwelling Older Adults: A Systematic Review*, Jan. 2016. DOI: 10.1016/j.jamda.2015.07.005.

[19] J. F. Guest, M. Panca, J. P. Baeyens, F. de Man, O. Ljungqvist, C. Pichard, S. Wait, and L. Wilson, "Health economic impact of managing patients following a community-based diagnosis of malnutrition in the UK", *Clinical Nutrition*, vol. 30, no. 4, pp. 422–429, Aug. 2011, ISSN: 02615614. DOI: 10.1016/j.clnu.2011.02.002.

[20] S. Khalatbari-Soltani and P. Marques-Vidal, "The economic cost of hospital malnutrition in Europe; a narrative review", *Clinical Nutrition ESPEN*, vol. 10, no. 3, e89–e94, 2015, ISSN: 24054577. DOI: 10.1016/j.clnesp.2015.04.003. [Online]. Available: http://dx.doi.org/10.1016/j.clnesp.2015.04.003.

[21] T. Cederholm, R. Barazzoni, P. Austin, P. Ballmer, G. Biolo, S. C. Bischoff, C. Compher, I. Correia, T. Higashiguchi, M. Holst, G. L. Jensen, A. Malone, M. Muscaritoli, I. Nyulasi, M. Pirlich, E. Rothenberg, K. Schindler, S. M. Schneider, M. A. de van der Schueren, C. Sieber, L. Valentini, J. C. Yu, A. Van Gossum, and P. Singer, "ESPEN guidelines on definitions and terminology of clinical nutrition", *Clinical Nutrition*, vol. 36, no. 1, pp. 49–64, 2017, ISSN: 15321983. DOI: 10.1016/j.clnu.2016.09.004.

[22] *Malnutrition symptoms & treatments - Illnesses & conditions | NHS inform*. [Online]. Available: https://www.nhsinform.scot/illnesses-and-conditions/nutritional/malnutrition.

[23] D. Ribeiro, J. Ribeiro, M. J. M. Vasconcelos, E. F. Vieira, and A. C. de Barros, "SousChef: Improved meal recommender system for Portuguese older adults", *Communications in Computer and Information Science*, vol. 869, pp. 107–126, 2018, ISSN: 18650929. DOI: 10.1007/978-3-319-93644-4{_}6.

[24] S. Mika, "Challenges for nutrition recommender systems", *CEUR Workshop Proceedings*, vol. 786, pp. 25–33, 2011, ISSN: 16130073.

[25] *Nutrition*. [Online]. Available: https://www.who.int/health-topics/nutrition.

[26] D. Volkert, A. M. Beck, T. Cederholm, A. Cruz-Jentoft, S. Goisser, L. Hooper, E. Kiesswetter, M. Maggio, A. Raynaud-Simon, C. C. Sieber, L. Sobotka, D. van Asselt, R. Wirth, and S. C. Bischoff, "ESPEN guideline on clinical nutrition and hydration in geriatrics", *Clinical Nutrition*, vol. 38, no. 1, pp. 10–47, 2019, ISSN: 15321983. DOI: 10.1016/j.clnu.2018.05.024. [Online]. Available: https://doi.org/10.1016/j.clnu.2018.05.024.

[27] *Demographics of Mobile Device Ownership and Adoption in the United States | Pew Research Center*. [Online]. Available: https://www.pewresearch.org/internet/fact-sheet/mobile/.

[28] *Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally | Pew Research Center*. [Online]. Available: https://www.pewresearch.org/global/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/.

[29] K. Singh and A. B. Landman, "Mobile Health", in *Key Advances in Clinical Informatics: Transforming Health Care through Health Information Technology*, Elsevier Inc., Jul. 2017, pp. 183–196, ISBN: 9780128095256. DOI: 10.1016/B978-0-12-809523-2.00013-3.

[30] "mHealth Economics 2017 – Current Status and Future Trends in Mobile Health",

[31] *Tracking for Health | Pew Research Center*. [Online]. Available: https://www.pewresearch.org/internet/2013/01/28/tracking-for-health/.

[32] *Millennials stand out for their technology use | Pew Research Center*. [Online]. Available: https://www.pewresearch.org/fact-tank/2019/09/09/us-generations-technology-use/.

[33] *Council Post: How AI Is Revolutionizing Health Care*. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2020/01/15/how-ai-is-revolutionizing-health-care/#5c12023a403c.

[34] D. Ribeiro, J. Machado, J. Ribeiro, M. J. M. Vasconcelos, E. F. Vieira, and A. C. De Barros, "SousChef: Mobile meal recommender system for older adults", *ICT4AWE 2017 - Proceedings of the 3rd International Conference on Information and Communication Technologies for Ageing Well and e-Health*, no. Ict4awe, pp. 36–45, 2017. DOI: 10.5220/0006281900360045.

[35] E. Van den Broeck, B. Zarouali, and K. Poels, "Chatbot advertising effectiveness: When does the message get through?", *Computers in Human Behavior*, vol. 98, pp. 150–157, Sep. 2019, ISSN: 07475632. DOI: 10.1016/j.chb.2019.04.009.

[36] P. B. Brandtzaeg and A. Følstad, "Why people use chatbots", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10673 LNCS, Springer Verlag, 2017, pp. 377–392, ISBN: 9783319702834. DOI: 10.1007/978-3-319-70284-1{_}30.

[37] S. S. Sundar, *Meida effects 2.0: Social and psychological effects of communication technologies*, 2009.

[38] S. S. Sundar and A. M. Limperos, "Uses and Grats 2.0: New Gratifications for New Media", *Journal of Broadcasting & Electronic Media*, vol. 57, no. 4, pp. 504–525, Oct. 2013, ISSN: 0883-8151. DOI: 10.1080/08838151.2013.845827. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/08838151.2013.845827.

[39] J. L. Z. Montenegro, C. A. da Costa, and R. da Rosa Righi, "Survey of conversational agents in health", *Expert Systems with Applications*, vol. 129, pp. 56–67, 2019, ISSN: 09574174. DOI: 10.1016/j.eswa.2019.03.054. [Online]. Available: https://doi.org/10.1016/j.eswa.2019.03.054.

[40] GUIDE Consortium, "User group definitions, end-user needs, requirement analysis and deployment guidelines", Tech. Rep., 2011. [Online]. Available: http://edream-h2020.eu/.

[41] *How chatbots could replace your HR department - BBC Worklife*. [Online]. Available: https://www.bbc.com/worklife/article/20180315-how-chatbots-could-replace-your-hr-department.

[42] A. Xu, Z. Liu, Y. Guo, V. Sinha, and R. Akkiraju, "A new chatbot for customer service on social media", in *Conference on Human Factors in Computing Systems - Proceedings*, vol. 2017-May, Association for Computing Machinery, May 2017, pp. 3506–3510, ISBN: 9781450346559. DOI: 10.1145/3025453.3025496.

[43] P. Wu and C. Miller, "Results from a field study: The need for an emotional relationship between the elderly and their assistive technologies", *Foundations of Augmented Cognition, Vol 11*, pp. 889–898, 2005.

[44] J. Feine, S. Morana, and U. Gnewuch, "Measuring Service Encounter Satisfaction with Customer Service Chatbots using Sentiment Analysis", *Wirtschaftsinformatik 2019 Proceedings*, Feb. 2019. [Online]. Available: https://aisel.aisnet.org/wi2019/track10/papers/2.

[45] L. Sobotka, Ed., *Basics in Clinical Nutrition*, 4th. Galen, 2012.

[46] "WHO | Global Nutrition Targets 2025: Stunting policy brief", *WHO*, 2018.

[47] "Body mass index - BMI", Jan. 2020.

[48] "Noncommunicable diseases prevention and control in the South-eastern Europe Health Network An analysis of intersectoral collaboration", Tech. Rep., 2012.

[49] World Health Organization, "World health statistics 2018: monitoring health for the SDGs, sustainable development goals", World Health Organization, Geneva, Switzerland, Tech. Rep., 2018. [Online]. Available: https://www.who.int/gho/publications/world_health_statistics/2018/en/.

[50] *Noncommunicable diseases*. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/noncommunicable-diseases.

[51] *Malnutrition NHS - NHS*. [Online]. Available: https://www.nhs.uk/conditions/malnutrition/.

[52] F. Branca, A. Lartey, S. Oenema, V. Aguayo, G. A. Stordalen, R. Richardson, M. Arvelo, and A. Afshin, "Transforming the food system to fight non-communicable diseases", *BMJ (Online)*, vol. 364, Jan. 2019, ISSN: 17561833. DOI: 10.1136/bmj.l296.

[53] E. Kesse-Guyot, S. Péneau, C. Méjean, F. Szabo de Edelenyi, P. Galan, S. Hercberg, and D. Lairon, "Profiles of Organic Food Consumers in a Large Sample of French Adults: Results from the Nutrinet-Santé Cohort Study", *PLoS ONE*, vol. 8, no. 10, Oct. 2013, ISSN: 19326203. DOI: 10.1371/journal.pone.0076998.

[54] *Food systems and diets: Facing the challenges of the 21st century*. 2016, ISBN: 9780995622807.

[55] *GBD Compare | IHME Viz Hub*. [Online]. Available: https://vizhub.healthdata.org/gbd-compare/.

[56] WHO, "WHO Fact sheet N°394 Healthy diet", Geneva, Switzerland, Tech. Rep., 2015. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs394/en/.

[57] World Health Organisation, "FAO/WHO Expert Consultation. Diet, nutrition and the prevention of chronic diseases: report of a joint WHO/FAO expert consultation, Geneva, 28 January", Tech. Rep. 2, 2003, pp. 61–71.

[58] World Health Organization, "WHO Guideline: Sugars intake for adults and children", *WHO Library Cataloguing-in-Publication Data*, vol. 26, no. 4, pp. 34–36, 2015, ISSN: 20354606. DOI: 9789241549028.

[59] B. Burlingame, C. Nishida, R. Uauy, and R. Weisell, "Fats and fatty acids in human nutrition: Introduction", Geneva, Switzerland, Tech. Rep. 1-3, 2009, pp. 5–7. DOI: 10.1159/000228993.

[60] WHO, "Sodium Intake for Adults and Children", Tech. Rep., 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23658998.

[61] European Parliament and Council, "Regulation (EU) No 1169/2011 of the European Parliamanent and Council of 25 October 2011 on the provision of food information to consumers (2011) Official Journal L304, 22.11.2011, p. 18–63", *Official Journal of the European Union, L 304, 22 November 2011*, vol. 54, pp. 18–63, 2011. DOI: 10.3000/19770677.L{_}2011.304.eng. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2011:304:TOC.

[62] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, *Recommendation systems: Principles, methods and evaluation*, Nov. 2015. DOI: 10.1016/j.eij.2015.06.005.

[63] S. H. Min and I. Han, "Detection of the customer time-variant pattern for improving recommender systems", *Expert Systems with Applications*, vol. 28, no. 2, pp. 189–199, Feb. 2005, ISSN: 09574174. DOI: 10.1016/j.eswa.2004.10.001.

[64] G. Adomavicius and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*, Jun. 2005. DOI: 10.1109/TKDE.2005.99.

[65] L. S. Chen, F. H. Hsu, M. C. Chen, and Y. C. Hsu, "Developing recommender systems with the consideration of product profitability for sellers", *Information Sciences*, vol. 178, no. 4, pp. 1032–1048, Feb. 2008, ISSN: 00200255. DOI: 10.1016/j.ins.2007.09.027.

[66] K. B. Cohen, *Biomedical Natural Language Processing and Text Mining*, Error. Elsevier Inc., 2013, pp. 141–177, ISBN: 9780124016781. DOI: 10.1016/B978-0-12-401678-1.00006-3. [Online]. Available: http://dx.doi.org/10.1016/B978-0-12-401678-1.00006-3.

[67] R. Ferreira, R. D. Lins, S. J. Simske, F. Freitas, and M. Riss, "Assessing sentence similarity through lexical, syntactic and semantic analysis", *Computer Speech and Language*, vol. 39, pp. 1–28, Sep. 2016, ISSN: 10958363. DOI: 10.1016/j.csl.2016.01.003.

[68] T. Pedersen, S. V. Pakhomov, S. Patwardhan, and C. G. Chute, "Measures of semantic similarity and relatedness in the biomedical domain", *Journal of Biomedical Informatics*, vol. 40, no. 3, pp. 288–299, Jun. 2007, ISSN: 15320464. DOI: 10.1016/j.jbi.2006.06.004.

[69] I. Bornkessel-Schlesewsky and M. Schlesewsky, "The Role of Prominence Information in the Real-Time Comprehension of Transitive Constructions: A Cross-Linguistic Approach", *Language and Linguistics Compass*, vol. 3, no. 1, pp. 19–58, Jan. 2009, ISSN: 1749818X. DOI: 10.1111/j.1749-818X.2008.00099.x. [Online]. Available: http://doi.wiley.com/10.1111/j.1749-818X.2008.00099.x.

[70] E. Cambria and B. White, *Jumping NLP curves: A review of natural language processing research*, 2014. DOI: 10.1109/MCI.2014.2307227.

[71] J. M. Gomez-Perez, D. Vila, R. Denaux, and C. Badenes, "Hybrid techniques for knowledge-based NLP: Knowledge graphs meet machine learning and all their friends", *CEUR Workshop Proceedings*, vol. 2065, pp. 69–70, 2017, ISSN: 16130073.

[72] M. Zhou, N. Duan, S. Liu, and H.-Y. Shum, "Progress in Neural NLP: Modeling, Learning, and Reasoning", *Engineering*, Jan. 2020, ISSN: 20958099. DOI: 10.1016/j.eng.2019.12.014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2095809919304928.

[73] F. A. Zaraket and A. Jaber, "MATAr: Morphology-based tagger for Arabic", in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, 2013, ISBN: 9781479907922. DOI: 10.1109/AICCSA.2013.6616418.

[74] M. Sabra and A. Alawieh, *Semiautomatic Annotator for Medical NLP Applications: About the Tool*. Elsevier Inc., 2019, pp. 143–150, ISBN: 9780128095560. DOI: 10.1016/b978-0-12-809556-0.00010-1. [Online]. Available: http://dx.doi.org/10.1016/B978-0-12-809556-0.00010-1.

[75] F. Y. Y. Choi, P. Wiemer-hastings, and J. Moore, "Latent semantic analysis for text segmentation", *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, vol. 102, pp. 109–117, 2001.

[76] C. D. Manning, H. Schütze, and G. Weikurn, "Foundations of Statistical Natural Language Processing", *SIGMOD Record*, vol. 31, no. 3, pp. 37–38, 2002, ISSN: 01635808. DOI: 10.1145/601858.601867.

[77] M. Hassler and G. Fliedl, "Text preparation through extended tokenization", in *WIT Transactions on Information and Communication Technologies*, vol. 37, 2006, pp. 13–21, ISBN: 1845641787. DOI: 10.2495/DATA060021.

[78] S. Sun, C. Luo, and J. Chen, "A review of natural language processing techniques for opinion mining systems", *Information Fusion*, vol. 36, pp. 10–25, 2017, ISSN: 15662535. DOI: 10.1016/j.inffus.2016.10.004. [Online]. Available: http://dx.doi.org/10.1016/j.inffus.2016.10.004.

[79] A. Goyal, V. Gupta, and M. Kumar, *Recent Named Entity Recognition and Classification techniques: A systematic review*, Aug. 2018. DOI: 10.1016/j.cosrev.2018.06.001.

[80] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface*. Cham: Springer International Publishing, 2016, ISBN: 978-3-319-32965-9. DOI: 10.1007/978-3-319-32967-3. [Online]. Available: http://link.springer.com/10.1007/978-3-319-32967-3.

[81] A. Taylor, M. Marcus, and B. Santorini, "The Penn Treebank: An Overview", in, 2003, pp. 5–22. DOI: 10.1007/978-94-010-0201-1{_}1.

[82] M. F. Porter, "An algorithm for suffix stripping", Tech. Rep.

[83] J. M. Moreira, A. C. P. L. F. de Carvalho, and T. Horváth, *A General Introduction to Data Analytics*. 2018, ISBN: 9781119296256. DOI: 10.1002/9781119296294.

[84] P. H. Chen, *Essential Elements of Natural Language Processing: What the Radiologist Should Know*, Jan. 2019. DOI: 10.1016/j.acra.2019.08.010.

[85] E. Delavenay and K. Delavenay, "An introduction to machine translation", 1960. [Online]. Available: https://ulti.in.net/1577490887.pdf.

[86] H. Liu, T. Christiansen, W. A. Baumgartner, and K. Verspoor, "BioLemmatizer: a lemmatization tool for morphological processing of biomedical text", *Journal of Biomedical Semantics*, vol. 3, no. 1, p. 3, 2012, ISSN: 2041-1480. DOI: 10.1186/2041-1480-3-3. [Online]. Available: http://jbiomedsem.biomedcentral.com/articles/10.1186/2041-1480-3-3.

[87] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008, p. 482, ISBN: 9780521865715.

[88] M.-C. De Marneffe and C. D. Manning, "Stanford typed dependencies manual", Tech. Rep., 2008. [Online]. Available: http:/www.universaldependencies.org.

[89] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", in *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, International Conference on Learning Representations, ICLR, 2013.

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

[91] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need", Tech. Rep.

[92] P. Wargnier, A. Malaisé, J. Jacquemot, S. Benveniste, P. Jouvelot, M. Pino, and A. S. Rigaud, "Towards attention monitoring of older adults with cognitive impairment during interaction with an embodied conversational agent", in *2015 3rd IEEE VR International Workshop on Virtual and Augmented Assistive Technology, VAAT 2015*, Institute of Electrical and Electronics Engineers Inc., Jul. 2015, pp. 23–28, ISBN: 9781467365185. DOI: 10.1109/VAAT.2015.7155406.

[93] J. Weizenbaum, "ELIZA-A Computer Program For the Study of Natural Language Communication Between Man and Machine", Tech. Rep. 1, 1966, pp. 36–71. [Online]. Available: http://i5.nyu.edu/~mm64/x52.9265/january1966.html.

[94] A. M. Turing, "Computing machinery and intelligence", in *Machine Intelligence: Perspectives on the Computational Model*, Taylor and Francis, Jan. 2012, pp. 1–28, ISBN: 9781136525049. DOI: 10.1093/mind/lix.236.433.

[95] S. Hussain, O. Ameri Sianaki, and N. Ababneh, "A Survey on Conversational Agents/Chatbots Classification and Design Techniques", in *Advances in Intelligent Systems and Computing*, vol. 927, Springer Verlag, 2019, pp. 946–956, ISBN: 9783030150341. DOI: 10.1007/978-3-030-15035-8{_}93.

[96] J. R. Bellegarda, "Large-Scale Personal Assistant Technology Deployment: the Siri Experience", Tech. Rep., 2013.

[97] A. V. Haridas, R. Marimuthu, and V. G. Sivakumar, "A critical review and analysis on techniques of speech recognition: The road ahead", *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 22, no. 1, pp. 39–57, 2018, ISSN: 18758827. DOI: 10.3233/KES-180374.

[98] L. Deng, J. Li, J. T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at Microsoft", Tech. Rep., 2013, pp. 8604–8608. DOI: 10.1109/ICASSP.2013.6639345.

[99] G. Hinton, L. Deng, D. Yu, G. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012, ISSN: 10535888. DOI: 10.1109/MSP.2012.2205597.

[100] F. Seide, G. Li, and D. Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks", Tech. Rep., 2011.

[101] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio", Tech. Rep., 2016. [Online]. Available: http://arxiv.org/abs/1609.03499.

[102] *WaveNet: A generative model for raw audio | DeepMind*. [Online]. Available: https://deepmind.com/blog/article/wavenet-generative-model-raw-audio.

[103] L. Wittgenstein, *Philosophical investigations (G. E. M. Anscombe, P. M. S. Hacker, J. Schulte, Trans.)* P. M. S. Hacker and J. Schulte, Eds. Wiley-Blackwell, 2009, ISBN: 9781405159289.

[104] J. L. (L. Austin, J. O. (O. Urmson, and M. Sbisà, *How to do things with words*. Oxford University Press, 1976, p. 169, ISBN: 9780192812056.

[105] J. R. Searle, *Speech Acts*. Cambridge University Press, Jan. 1969, ISBN: 9780521096263. DOI: 10.1017/CBO9781139173438. [Online]. Available: https://www.cambridge.org/core/product/identifier/9781139173438/type/book.

[106] S. Palmer, R. Woolfe, J. Martin, and F. Margison, *The Conversational Model*. 2014, pp. 58–73, ISBN: 9783319329659. DOI: 10.4135/9781446280409.n3.

[107] J. Allen, *Natural language understanding*. Benjamin/Cummings Pub. Co, 1995, p. 654, ISBN: 9780805303346.

[108] T. Kï, H. Uszkoreit, and F. Xu, "Using Syntactic and Semantic based Relations for Dialogue Act Recognition", Tech. Rep., 2010, pp. 570–578. [Online]. Available: http://www.metaversum.com/.

[109] N. Webb and T. Liu, "Investigating the Portability of Corpus-Derived Cue Phrases for Dialogue Act Classification", Tech. Rep., 2008, pp. 977–984.

[110] D. Verbree, R. Rienks, and D. Heylen, "Dialogue-act tagging using smart feature selection; results on multiple corpora", in *2006 IEEE ACL Spoken Language Technology Workshop, SLT 2006, Proceedings*, 2006, pp. 70–73, ISBN: 1424408733. DOI: 10.1109/SLT.2006.326819.

[111] S. Keizer and R. O. Den Akker, "Dialogue act recognition under uncertainty using Bayesian networks", *Natural Language Engineering*, vol. 13, no. 4, pp. 287–316, Dec. 2007, ISSN: 13513249. DOI: 10.1017/S1351324905004067.

[112] M. Nagata and T. Morimoto, "First steps towards statistical modeling of dialogue to predict the speech act type of the next utterance", *Speech Communication*, vol. 15, no. 3-4, pp. 193–203, 1994, ISSN: 01676393. DOI: 10.1016/0167-6393(94)90071-X.

[113] J. Romportl and J. Matoušek, "Formal prosodic structures and their application in NLP", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3658 LNAI, 2005, pp. 371–378, ISBN: 3540287892. DOI: 10.1007/11551874{_}48.

[114] W. L. Wu, R. Z. Lu, J. Y. Duan, H. Liu, F. Gao, and Y. Q. Chen, "Spoken language understanding using weakly supervised learning", *Computer Speech and Language*, vol. 24, no. 2, pp. 358–382, Apr. 2010, ISSN: 08852308. DOI: 10.1016/j.csl.2009.05.002.

[115] L. Ni, C. Lu, N. Liu, and J. Liu, "MANDY: Towards a smart primary care chatbot application", in *Communications in Computer and Information Science*, vol. 780, Springer Verlag, 2017, pp. 38–52, ISBN: 9789811069888. DOI: 10.1007/978-981-10-6989-5{_}4.

[116] E. B. Dündar, T. C. ¸. Ekiç, O. Deniz, and S. Arslan, "A Hybrid Approach to Question-answering for a Banking Chatbot on Turkish: Extending Keywords with Embedding Vectors", Tech. Rep.

[117] S. Arsovski, H. Osipyan, M. I. Oladele, and A. D. Cheok, "Automatic knowledge extraction of any Chatbot from conversation", *Expert Systems with Applications*, vol. 137, pp. 343–348, Dec. 2019, ISSN: 09574174. DOI: 10.1016/j.eswa.2019.07.014.

[118] V. Kasinathan, F. S. Xuan, M. H. A. Wahab, and A. Mustapha, "Intelligent Healthcare Chatterbot (HECIA): Case study of medical center in Malaysia", in *2017 IEEE Conference on Open Systems, ICOS 2017*, vol. 2018-January, Institute of Electrical and Electronics Engineers Inc., Feb. 2018, pp. 32–37, ISBN: 9781538607312. DOI: 10.1109/ICOS.2017.8280270.

[119] J. Cahn, "CHATBOT: Architecture, Design, & Development", Tech. Rep., 2017.

[120] Z. Yan, N. Duan, J. Bao, P. Chen, M. Zhou, Z. Li, and J. Zhou, "DocChat: An information retrieval approach for chatbot engines using unstructured documents", in *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, vol. 1, Association for Computational Linguistics (ACL), 2016, pp. 516–525, ISBN: 9781510827585. DOI: 10.18653/v1/p16-1049.

[121] A. Lommatzsch and J. Katins, "An Information Retrieval-based Approach for Building Intuitive Chatbots for Large Knowledge Bases", Tech. Rep. [Online]. Available: https://dialogflow.com/.

[122] M. Henderson, I. Vuli´cvuli´c, D. Gerz, I. Casanueva, P. Budzianowski, S. Coope, G. Spithourakis, T.-H. Wen, N. M. Mrkši´c, and P.-H. Su, "Training Neural Response Selection for Task-Oriented Dialogue Systems", Tech. Rep., pp. 5392–5404.

[123] J.-C. Gu, T. Li, Q. Liu, X. Zhu, Z.-H. Ling, Z. Su, and S. Wei, "Speaker-Aware BERT for Multi-Turn Response Selection in Retrieval-Based Chatbots", Tech. Rep.

[124] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation", in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, Association for Computational Linguistics (ACL), 2014, pp. 1724–1734, ISBN: 9781937284961. DOI: 10.3115/v1/d14-1179.

[125] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation", in *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, Association for Computational Linguistics (ACL), 2016, pp. 1192–1202, ISBN: 9781945626258. DOI: 10.18653/v1/d16-1127.

[126] V. Rieser and O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems*. Springer Berlin Heidelberg, 2011. DOI: 10.1007/978-3-642-24942-6.

[127] A. Wang and K. Cho, "BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model", Tech. Rep. [Online]. Available: https://github.com/huggingface/.

[128] S. A. Abdul-Kader and J. Woods, "Survey on chatbot design techniques in speech conversation systems", *International Journal of Advanced Computer Science and Applications*, 2015.

[129] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing", *International Journal of Human - Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1995, ISSN: 10959300. DOI: 10.1006/ijhc.1995.1081.

[130] L. Bradeško and D. Mladenić, "A Survey of Chabot Systems through a Loebner Prize Competition", Tech. Rep., 2012. [Online]. Available: http://www-ai.ijs.si/eliza-cgi-bin/eliza_script.

[131] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The Microsoft 2017 Conversational Speech Recognition System", Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.06073.

[132] *Agents | Dialogflow Documentation | Google Cloud*. [Online]. Available: https://cloud.google.com/dialogflow/docs/agents-overview.

[133] *Intents | Dialogflow Documentation | Google Cloud*. [Online]. Available: https://cloud.google.com/dialogflow/docs/intents-overview.

[134] *Entities | Dialogflow Documentation | Google Cloud*. [Online]. Available: https://cloud.google.com/dialogflow/docs/entities-overview.

[135] *Contexts | Dialogflow Documentation | Google Cloud*. [Online]. Available: https://cloud.google.com/dialogflow/docs/contexts-overview.

[136] *Follow-up intents | Dialogflow Documentation | Google Cloud*. [Online]. Available: https://cloud.google.com/dialogflow/docs/contexts-follow-up-intents.

[137] *Dialogflow basics | Dialogflow Documentation | Google Cloud*. [Online]. Available: https://cloud.google.com/dialogflow/docs/basics.

[138] K. Papineni, S. Roukos, T. Ward, W.-j. Zhu, and Y. Heights, "IBM Research Report Bleu : a Method for Automatic Evaluation of Machine Translation", Tech. Rep., 2001, pp. 1–10. DOI: 10.3115/1073083.1073135. [Online]. Available: http://dl.acm.org/citation.cfm?id=1073135.

[139] S. Banerjee and A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments", Tech. Rep.

[140] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries", Tech. Rep.

[141] L. Nio, S. Sakti, G. Neubig, T. Toda, and S. N. Naist, "A Dialog System with Human-to-Human Conversation Example", Tech. Rep., pp. 41–42. [Online]. Available: http://wordnet.princeton.edu/.

[142] C.-W. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation", Tech. Rep.

[143] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems", Jun. 2015. [Online]. Available: http://arxiv.org/abs/1506.08909.

[144] M. Galley, C. Brockett, A. Sordoni, Y. Ji, M. Aim, C. Quirk, M. Mitchell, J. Gao, and B. Dolan, "ΔbLEU: A discriminative metric for generation tasks with intrinsically diverse targets", in *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, vol. 2, Association for Computational Linguistics (ACL), 2015, pp. 445–450, ISBN: 9781941643730. DOI: 10.3115/v1/p15-2073.

[145] S. Dutta, "Saarland University Department of Computational Linguistics Block Project : Neural Networks Software Project Evaluating a neural multi-turn chatbot using BLEU score Author :", no. April, 2019. DOI: 10.13140/RG.2.2.13374.84808.

[146] Y. Graham, N. Mathur, and T. Baldwin, "Accurate Evaluation of Segment-level Machine Translation Metrics", Tech. Rep., pp. 1183–1191. [Online]. Available: https://github.com/ygraham/.

[147] R. Lowe, M. Noseworthy, I. V. Serban, N. A-Gontier, Y. Bengio, and J. Pineau, "Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses", Tech. Rep. [Online]. Available: https://github.com/mike-n-7/ADEM..

[148] C. Tao, L. Mou, D. Zhao, and R. Yan, "RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems", Tech. Rep. [Online]. Available: http://www.msxiaoice.com/.

[149] A. Kannan, G. Brain, O. Vinyals, and G. Deepmind, "Adversarial Evaluation of Dialogue Models", Tech. Rep., 2017.

[150] A. S. Lokman and M. A. Ameedeen, "Modern chatbot systems: A technical review", in *Advances in Intelligent Systems and Computing*, vol. 881, Springer Verlag, 2019, pp. 1012–1023, ISBN: 9783030026820. DOI: 10.1007/978-3-030-02683-7{_}75.

[151] J. Brooke, "SUS-A quick and dirty usability scale", Tech. Rep.

[152] R. Likert, "A technique for the measurement of attitudes.", *Archives of Psychology*, vol. 22 140, p. 55, 1932.

[153] J. R. Lewis and J. Sauro, "The Factor Structure of the System Usability Scale", Tech. Rep.

[154] F. F. Reichheld, "The One Number You Need to Grow", Tech. Rep., 2003. [Online]. Available: www.hbr.org.

[155] B. Laugwitz, T. Held, and M. Schrepp, "Construction and Evaluation of a User Experience Questionnaire", in, 2008, pp. 63–76. DOI: 10.1007/978-3-540-89350-9{_}6. [Online]. Available: http://link.springer.com/10.1007/978-3-540-89350-9_6.

[156] I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, S. Rajeshwar, A. De Brebisson, J. M. R. Sotelo, D. Suhubdy, V. Michalski, A. Nguyen, J. Pineau, and Y. Bengio, "A Deep Reinforcement Learning Chatbot", Tech. Rep. [Online]. Available: www.alicebot.org..

[157] W. L. Johnson, C. Labore, and Y.-C. Chiu, "A Pedagogical Agent for Psychosocial Intervention on a Handheld Computer", Tech. Rep., 2004.

[158] E. Hudlicka, "Virtual training and coaching of health behavior: Example from mindfulness meditation training", *Patient Education and Counseling*, vol. 92, no. 2, pp. 160–166, Aug. 2013, ISSN: 07383991. DOI: 10.1016/j.pec.2013.05.007.

[159] L. Ring, T. Bickmore, and P. Pedrelli, "Real-Time Tailoring of Depression Counseling by Conversational Agent", *Iproceedings*, vol. 2, no. 1, e27, Dec. 2016, ISSN: 2369-6893. DOI: 10.2196/iproc.6065.

[160] A. ". Skip, ". Rizzo, P. Kenny, and T. D. Parsons, "Intelligent Virtual Patients for Training Clinical Skills", Tech. Rep. 3, 2011. [Online]. Available: http://www.dipp.nrw.de/..

[161] S. Tokunaga, H. Horiuchi, K. Tamamizu, S. Saiki, M. Nakamura, and K. Yasuda, "Deploying service integration agent for personalized smart elderly care", in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science, ICIS 2016 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Aug. 2016, ISBN: 9781509008063. DOI: 10.1109/ICIS.2016.7550873.

[162] A. C. King, T. W. Bickmore, M. I. Campero, L. A. Pruitt, and J. L. Yin, "Employing Virtual Advisors in Preventive Care for Underserved Communities: Results From the COMPASS Study", *Journal of Health Communication*, vol. 18, no. 12, pp. 1449–1464, Dec. 2013, ISSN: 1081-0730. DOI: 10.1080/10810730.2013.798374. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10810730.2013.798374.

[163] Á. Alesanco, J. Sancho, Y. Gilaberte, E. Abarca, and J. García, "Bots in messaging platforms, a new paradigm in healthcare delivery: Application to custom prescription in dermatology", in *IFMBE Proceedings*, vol. 65, Springer Verlag, 2017, pp. 185–188, ISBN: 9789811051210. DOI: 10.1007/978-981-10-5122-7{_}47.

[164] M. Hirano, K. Ogura, M. Kitahara, D. Sakamoto, and H. Shimoyama, "Designing behavioral self-regulation application for preventive personal mental healthcare", *Health Psychology Open*, vol. 4, no. 1, p. 205 510 291 770 718, Jan. 2017, ISSN: 2055-1029. DOI: 10.1177/2055102917707185. [Online]. Available: http://journals.sagepub.com/doi/10.1177/2055102917707185.

[165] A. Bresó, J. Martínez-Miranda, C. Botella, R. M. Baños, and J. M. García-Gómez, "Usability and acceptability assessment of an empathic virtual agent to prevent major depression", *Expert Systems*, vol. 33, no. 4, pp. 297–312, Aug. 2016, ISSN: 02664720. DOI: 10.1111/exsy.12151. [Online]. Available: http://doi.wiley.com/10.1111/exsy.12151.

[166] M. Turunen, J. Hakulinen, O. Ståhl, B. Gambäck, P. Hansen, M. C. Rodríguez Gancedo, R. S. De La Cámara, C. Smith, D. Charlton, and M. Cavazza, "Multimodal and mobile conversational Health and Fitness Companions", *Computer Speech and Language*, vol. 25, no. 2, pp. 192–209, Apr. 2011, ISSN: 08852308. DOI: 10.1016/j.csl.2010.04.004.

[167] L. Dybkjær, N. O. Bernsen, and W. Minker, "Evaluation and usability of multimodal spoken language dialogue systems", *Speech Communication*, vol. 43, no. 1-2, pp. 33–54, Jun. 2004, ISSN: 01676393. DOI: 10.1016/j.specom.2004.02.001.

[168] A. Fadhil, "Can a Chatbot Determine My Diet?: Addressing Challenges of Chatbot Application for Meal Recommendation", 2018. [Online]. Available: http://arxiv.org/abs/1802.09100.

[169] A. A. Abd-alrazaq, M. Alajlani, A. A. Alalwan, B. M. Bewick, P. Gardner, and M. Househ, *An overview of the features of chatbots in mental health: A scoping review*, Dec. 2019. DOI: 10.1016/j.ijmedinf.2019.103978.

[170] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, "Slot-Gated Modeling for Joint Slot Filling and Intent Prediction", Tech. Rep., pp. 753–757. [Online]. Available: https://github.com/.

[171] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "DIET: Lightweight Language Understanding for Dialogue Systems", Apr. 2020. [Online]. Available: http://arxiv.org/abs/2004.09936.

[172] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding", Apr. 2018. [Online]. Available: http://arxiv.org/abs/1804.07461.

[173] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems", May 2019. [Online]. Available: http://arxiv.org/abs/1905.00537.

[174] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations", in *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, Association for Computational Linguistics (ACL), Feb. 2018, pp. 2227–2237, ISBN: 9781948087278. DOI: 10.18653/v1/n18-1202.

[175] A. R. Openai, K. N. Openai, T. S. Openai, and I. S. Openai, "Improving Language Understanding by Generative Pre-Training", Tech. Rep., 2018. [Online]. Available: https://gluebenchmark.com/leaderboard.

[176] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-Reconstruction Learning for Open-Set Recognition", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 4011–4020, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1812.04246.

[177] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie, "A Hierarchical Recurrent Encoder-Decoder For Generative Context-Aware Query Suggestion", *Ccs '15*, pp. 158–169, Jul. 2015. [Online]. Available: http://arxiv.org/abs/1507.02221.

[178] S. Sahay, S. H. Kumar, E. Okur, H. Syed, and L. Nachman, "Modeling Intent, Dialog Policies and Response Adaptation for Goal-Oriented Interactions", Tech. Rep. [Online]. Available: https://tfhub.dev/google/universal-sentence-encoder-.

[179] V. Vlasov, J. E. M. Mosig, and A. Nichol, "Dialogue Transformers", Oct. 2019. [Online]. Available: http://arxiv.org/abs/1910.00486.

[180] V. Vlasov, A. Drissner-Schmid, and A. Nichol, "Few-Shot Generalization Across Dialogue Tasks", Nov. 2018. [Online]. Available: http://arxiv.org/abs/1811.11707.

[181] *Instituto dos Registos e Notariado: Nomes próprios de cidadãos portugueses nos últimos 3 anos*. [Online]. Available: https://www.irn.mj.pt/sections/irn/a_registral/registos-centrais/docs-da-nacionalidade/vocabulos-admitidos-e/.

[182] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation", Tech. Rep.

[183] T. Ohlsson, *Encyclopedia of Food Sciences and Nutrition*, 2003. DOI: 10.1016/B0-12-227055-X/01335-3. [Online]. Available: http://dx.doi.org/10.1016/B0-12-227055-X/01335-3.

[184] C. J. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text", Tech. Rep., 2014. [Online]. Available: http://sentic.net/.

[185] G. Miller and Princeton University. Cognitive Science Laboratory., *WordNet*. MIT Press, 1998, ISBN: 9780262561167.

[186] R. Speer, J. Chin, and C. Havasi, "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge", Dec. 2016. [Online]. Available: http://arxiv.org/abs/1612.03975.

[187] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "StarSpace: Embed all the things!", in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, AAAI press, Sep. 2018, pp. 5569–5577, ISBN: 9781577358008.

[188] W. Tansey, E. W. Lowe, and J. G. Scott, "Diet2Vec: Multi-scale analysis of massive dietary data", Tech. Rep. [Online]. Available: http://www.loseit.com/.

[189] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting Word Vectors to Semantic Lexicons", Tech. Rep. [Online]. Available: http://nlp.stanford.edu/.

[190] J. F. Kelley, "An iterative design methodology for user-friendly natural language office information applications", *ACM Transactions on Information Systems (TOIS)*, vol. 2, no. 1, pp. 26–41, Jan. 1984, ISSN: 15582868. DOI: 10.1145/357417.357420. [Online]. Available: http://dl.acm.org/doi/10.1145/357417.357420.

[191] S. Holmes, A. Moorhead, R. Bond, H. Zheng, V. Coates, and M. McTear, "Usability testing of a healthcare chatbot: Can we use conventional methods to assess conversational user interfaces?", in *ECCE 2019 - Proceedings of the 31st European Conference on Cognitive Ergonomics: "Design for Cognition"*, New York, New York, USA: Association for Computing Machinery, Inc, Sep. 2019, pp. 207–214, ISBN: 9781450371667. DOI: 10.1145/3335082.3335094. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3335082.3335094.

[192] A. Bangor, P. Kortum, and J. Miller, "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale", Tech. Rep., 2009, pp. 114–123.

[193] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale", *International Journal of Human-Computer Interaction*, vol. 24, no. 6, pp. 574–594, Aug. 2008, ISSN: 10447318. DOI: 10.1080/10447310802205776. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/10447310802205776.

[194] *MeasuringU: Predicting Net Promoter Scores from System Usability Scale Scores*. [Online]. Available: https://measuringu.com/nps-sus/.

[195] M. Schrepp, A. Hinderks, and J. Thomaschewski, "Construction of a Benchmark for the User Experience Questionnaire (UEQ)", *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 4, p. 40, 2017, ISSN: 1989-1660. DOI: 10.9781/ijimai.2017.445.

Appendix A

Complementary Information

In this chapter some complementary information and results are presented.

A.1 Evaluation Questionnaire – Google Form

This section includes the Google Form developed for data collection. It is possible to observe the several requests made to the volunteers

Chatbot for a Nutritional Recommender System

** TODAS AS RESPOSTAS DEVEM SER ESCRITAS EM INGLÊS. OBRIGADO**
[PT] No âmbito da dissertação da tese de Mestrado, estou a construir um chatbot para facilitar na navegação e interação com um sistema de planeamento alimentar.
Um chatbot é um assistente virtual do género da Siri ou do Google Assistant que temos nos nossos telemóveis. Podemos dar-lhes comandos por voz ou por texto.
O objetivo deste form é recolher dados para conseguir validar o meu modelo. Assim, serão apresentadas algumas tarefas que o chatbot terá de conseguir fazer e ser-vos-á pedido que escrevam um exemplo do que diriam ao chatbot para cada uma. Sejam criativos (não em demasia) mas pensem como é que interagiam com um chatbot! Serão apresentados com 2/3 exemplos por intenção que são ilustrativos das infinitas possibilidades e combinações.

ALL ANSWERS SHOULD BE IN ENGLISH. THANK YOU!
[EN] Under the scope of my Masters Thesis dissertation, I am building a chatbot to replace and ease the navigation and interaction with a meal planning system.
A chatbot is a virtual assistant like Siri or Google Assistant that we all have in our smartphones. You can give commands in the form of text or voice.
The goal of this form is to gather data in order to validate my system and would very much appreciate your help. Therefore, I will present to you some tasks that the chatbot should be able to perform. I'll ask you to write an example of what you would say to the chatbot to get that done. Be creative (not too much) and think how you would interact with a chatbot! You'll be presented with 2/3 examples to illustrate the infinite number of possibilities and combinations.

Next

Chatbot for a Nutritional Recommender System

* Required

Create a plan

The first task is to ask the chatbot to create a meal plan. At this stage you don't have a plan yet, so don't ask the bot to show you one.
You can ask him to create a weekly plan or a daily plan. You can ask him to create it for the time period you want, next Monday, weekend, next week, 30th of may...

The goal is to write an example that shows that you want to generate a meal plan, the time is up to you.

ALL ANSWERS SHOULD BE IN ENGLISH. THANK YOU!

Here are several examples:
I'd like you to create me a plan for next Monday!
Hey bot, create me a plan for this week please.
I need a plan for the weekend!

Now write your own example *

Your answer

Now write your own example *

Your answer

Back Next

Chatbot for a Nutritional Recommender System

* Required

Ask to see a plan

Now you can assume that the plan is already created.
The second task is to see a plan for a given time frame: a week, a day, a specific meal, or even a part of it like the main dish, the drink, the dessert, the soup or the fruit.

The goal is to write an example that shows that you want to see a meal plan, the time, meal and part is up to you. You can include them or not.

ALL ANSWERS SHOULD BE IN ENGLISH. THANK YOU!
Here are several examples:
Show me the meals for Monday dinner and Tuesday lunch!
What do you have for me for this week?
What is the drink for sunday's afternoon snack?
What will I eat for soup?
What is my next meal?

Now write your own example *

Your answer

Now write your own example *

Your answer

Now write your own example *

Your answer

Your answer

Page 4 of 9

Back Next

Never submit passwords through Google Forms.

Chatbot for a Nutritional Recommender System

* Required

Express food preference

Seeing the plan, you can dislike or love one ingredient or meal.
You should let your bot know what you like and dislike in order to create better recommendations.
In this case, I'll ask you to write the ingredient or recipe that you said in a separate box as well. Please do not repeat the ingredients!
As a third task I'll also ask you to rate the preference that you gave from 0-5 according to what you wrote.
The higher the value the more you expressed your positivity.
The lower the value, the more you expressed your negative feeling.
The goal is to rate the intensity of the sentence and not the ingredient.
Example: Strawberries are my favourite food in the world! - 5
Example 2: I hate strawberries! Please don't ever create a plan with them! - 0
Example 3: I like strawberries - 3

ALL ANSWERS SHOULD BE IN ENGLISH. THANK YOU!

Example: I really like anchovies! Now write your own example *

Your answer

Here you write the ingredient/meal. The example: anchovies *

Your answer

Rate the sentence feeling here from 0-5. According to the example I'd answer 4 *

- 0
- 1
- 2
- 3
- 4
- 5

Example 2: I hate every meal that has smoked salmon in it! Now write your own example *

Your answer

Here you write the ingredient/meal. The example: smoked salmon *

Your answer

Rate the sentence feeling here from 0-5. According to the example I'd answer 1 *

- 0
- 1
- 2
- 3
- 4

4

5

Example 3: Chocolate milk is awful! I hate it! Now write your own example *

Your answer

Here you write the ingredient/meal. The example: chocolate milk *

Your answer

Rate the sentence feeling here from 0-5. According to the example I'd answer 0 *

0

1

2

3

4

5

Page 5 of 9

Back Next

Chatbot for a Nutritional Recommender System

* Required

Express food preference

Seeing the plan, you can dislike or love one ingredient or meal.
You should let your bot know what you like and dislike in order to create better recommendations.
In this case, I'll ask you to write the ingredient or recipe that you said in a separate box as well. Please do not repeat the ingredients!
As a third task I'll also ask you to rate the preference that you gave from 0-5 according to what you wrote.
The higher the value the more you expressed your positivity.
The lower the value, the more you expressed your negative feeling.
The goal is to rate the intensity of the sentence and not the ingredient.
Example: Strawberries are my favourite food in the world! - 5
Example 2: I hate strawberries! Please don't ever create a plan with them! - 0
Example 3: I like strawberries - 3

ALL ANSWERS SHOULD BE IN ENGLISH. THANK YOU!

Example: I really like anchovies! Now write your own example *

Your answer

Here you write the ingredient/meal. The example: anchovies *

Your answer

Rate the sentence feeling here from 0-5. According to the example I'd answer 4 *

- 0
- 1
- 2
- 3
- 4
- 5

Example 2: I hate every meal that has smoked salmon in it! Now write your own example *

Your answer

Here you write the ingredient/meal. The example: smoked salmon *

Your answer

Rate the sentence feeling here from 0-5. According to the example I'd answer 1 *

- 0
- 1
- 2
- 3
- 4

4

5

Example 3: Chocolate milk is awful! I hate it! Now write your own example *

Your answer

Here you write the ingredient/meal. The example: chocolate milk *

Your answer

Rate the sentence feeling here from 0-5. According to the example I'd answer 0 *

0

1

2

3

4

5

Page 5 of 9

Back Next

Chatbot for a Nutritional Recommender System

* Required

Make a change in the plan

Imagine the bot shows you some meals you don't like or you'd rather replace. You'll ask the bot for some alternatives. You can include meals or parts. You can also ask for ingredients in the replacements

The goal is to write an example that shows that you want to change something in the meal plan

ALL ANSWERS SHOULD BE IN ENGLISH. THANK YOU!

Example: I'd like to replace the dessert in this meal for something with chocolate! Now write your own example *

Your answer

If you mentioned some ingredient please write it down here. I mentioned 'chocolate' so I'd write down chocolate here. Otherwise you can ignore this question

Your answer

Example: Can you show me some substitutes for the lunch main dish? Now write your own example *

Your answer

If you mentioned some ingredient please write it down here. Otherwise you can ignore this question

Your answer

Page 6 of 9

Back Next

Never submit passwords through Google Forms.

Chatbot for a Nutritional Recommender System

* Required

Personal information

The bot must know some things about you. In this section I'll ask you 2 different commands. In the first one you should ask for the information that the bots knows about you. In the second one you'll give the bot some kind of personal information. You can ask/tell your name, age, birth date, weight, height, food restrictions (vegan, vegetarian, diabetic...), activity level and gender.

Ask for information that is saved in the bot's memory.
Examples for the first task:
I'd like to know what information you have about me!
Do you know when my birthday is?
Say my name!

Give personal information to the bot.
Examples for the second task:
I weight 85 kg and I am 1.82 meters tall!
My name is John Appleseed.
I'm 27 years old, bot!
I am very very active during the week

Now write your own example: 1st task *

Your answer

Now write your own example: 2nd task *

Your answer

Chatbot for a Nutritional Recommender System

* Required

Nutritional Composition

This is the last section.
Here you should ask the bot for nutritional information. You can ask for a specific meal, meal part, ingredient, time frame or for a recipe
You can also ask for specific nutritional concepts (energy, protein, fat, carbs, salt, water, vitamins...)

Examples:
How much protein and fat does the lunch contain?
What is the nutritional content of my meals for today?
What is the caloric content of this dessert?

Now write your own example *

Your answer

Now write your own example *

Your answer

A.2 SVM parameters for grid search

Table A.1: SVM parameters for grid search optimization. The underlined values are the ones used

Parameter	Value
C	1, 2, 5, 10, 20, 100
Kernel	linear, rbf
Gamma	0.1, 0.01
Scoring function	F1 weighted
Cross-Validation Folds	5

A.3 DIET classifier hyperparameters

Table A.2: Parameters and Hyperparameters used in DIET Classifier

Parameter / Hyperparameters	Value
Epochs	125 with early stopping
Number of transformer layers	2
Number of attention heads per transformer layer	4
Batch Size	[64; 256] increasing linearly per epoch
Learning rate	0.001
Embedding size	20
Loss	Softmax
Dropout Feedforward layers	0.2
Validation set	10%

A.4 LanguaL categories

Table A.3: Possible categories for each LanguaL facet

LanguaL facet	Categories	Number of Categories
A – Product Type	vegetables, excluding potatoes; cheese; fruits; fruit juices; pasta; fish and seafood; miscellaneous foods; pulses; bakery products; bread and rolls; breakfast cereals; red meat and meat products; sugar products, excluding chocolate; wine; coffee, tea, cocoa powder; starchy roots and potatoes; other alcoholic beverages; vegetable oils; rice and other cereal products; nuts; other milk products; milk; beer; margarine and lipids of mixed origin; eggs; poultry and poultry products; butter and animal fats; chocolate; sugar; non-alcoholic beverages; offals; flour	32
B – Food Source	carrot; cow; pineapple; pear; orange; animal (mammal); durum wheat; tomato; lemon sole; pea; date; mustard; haddock; citrus family; soybean; wheat; rye; chicken; sugar-producing plant; grain; rhubarb; cattle; broccoli; cucumber; onion; grape; cacao; potato; common kaki; plum; walnut; black currant; mung bean; atlantic mackerel; zucchini; fruit-producing plant; rice; swine; rosemary; pecan; lamb; coconut palm; vegetable corn, yellow; field corn; barley; red raspberry; field mushroom; rutabaga; oil-producing plant; green bell pepper; brussels sprout; garbanzo bean; hen; prune; vegetable-producing plant; lemon; red cabbage; cauliflower; green bean; olive; tuna; avocado; peanut; skipjack tuna; ewe; litchi; apricot; phaseolus vulgaris, edible seed cultivar; plaice; plant used as food source; clementine; palm; sage; strawberry; plant used for producing extract or concentrate; broad bean; plantain (musa); turkey (poultry); nut producing plant; cherry; wels catfish; water; romaine; fish, bony; spice or flavor-producing plant; melon; brazil nut; common oat; shrimp; atlantic cod; thyme; animal used as food source; sesame; mandarin orange; mushroom; nectarine; yam, tropical; watercress; rainbow trout; edible currant; red bell pepper; coriander; atlantic salmon; grape, european; cabbage; atlantic herring; oregano; peach; almond; cinnamon; american cranberry; european whiting; grapefruit; safflower; european sardine; popcorn; hungarian wax pepper; tea; kiwifruit; parsnip; food source not known; vegetable marrow; yeast; argentine anchovy; radish; starch-producing plant; lima bean; miniature tomato; sheep; asparagus; horseradish; red kidney bean; coffee; celery; hot pepper; butterhead lettuce; basil, sweet; edible rock crab; spearmint; pepper, white; papaya; satsuma orange; savoy cabbage; beet; pepper, black; eggplant; macadamia; french plantain; sodium hydrogen carbonate; cattle and swine; fish or lower water animal; watermelon; blackeyed pea; apple; squid; snow pea; peanut with other nut or seed; turnip; lentil; gooseberry; scallion; green olive; mango; cashew; fig; sugar maple; sunflower; halibut; calf; pomegranate; berry; common prawn; black gram bean; leek; duck; kale; pistachio; norway lobster; fruit used as vegetable; passion fruit; herring family; spinach; pea, edible seed cultivars; fungus; european lobster; lettuce; sodium chloride; damson plum; grape, sultana; sweet potato; european filbert; parsley; nutmeg; doe (goat); florence fennel; garlic; dill; cod; european blackberry; iceberg lettuce; adzuki bean; pheasant; stone pine; eel; saithe; protein-producing plant; rape	207
C – Part of Plant or Animal	root, tuber or bulb, without peel; curd; fruit, peel removed, core, pit or seed removed; fruit; seed, skin removed, germ present; fruit or seed; skeletal meat part, without bone or shell; seed, skin present, germ present; fruit, peel present, core, pit or seed removed; seed; skeletal meat part, without bone, without skin; fruit, peel only; whole egg without shell; sugar syrup or syrup solids; stem or stalk (without leaves); plant above surface, excluding fruit and seed; milk; sugar; fruit, peel present, core, pit or seed present; extract, concentrate or isolate of plant or animal; fat or oil; germinated or sprouted seed; skeletal meat part, without bone, with skin; juice; skeletal meat part, without bone and skin, with separable fat; leaf; skeletal meat part, without bone and skin, without separable fat; nut milk; milk or milk component; seed on cob, with or without husk; light whipping cream; head (plant); part of plant; cream; floret or flower; pod containing small, immature seed; light cream; fruit, peel undetermined, core, pit or seed removed; skeletal meat part; butter; starch; honey; fruit, peel removed, core, pit or seed present; part of plant or animal not applicable; tongue; kidney; cream or cream component; blood; liver; seed, skin removed, germ removed (endosperm); seed oil; buttermilk; part of algae or fungus; brown sugar; fruit, peel present; heavy cream; bark; pod or seed; sucrose; skin, animal; part of plant or animal not known; whole animal or most parts used; root, tuber or bulb, with peel; root; fat, trim; spear or shoot; white sugar; root, stem, leaf or flower; bran; protein extract, concentrate or isolate; meat part; tuber; bulb; whole plant or most parts used; organ meat; root, tuber or bulb, with part of top; skeletal meat part, with bone or shell; skeletal meat part, with bone, with skin	78

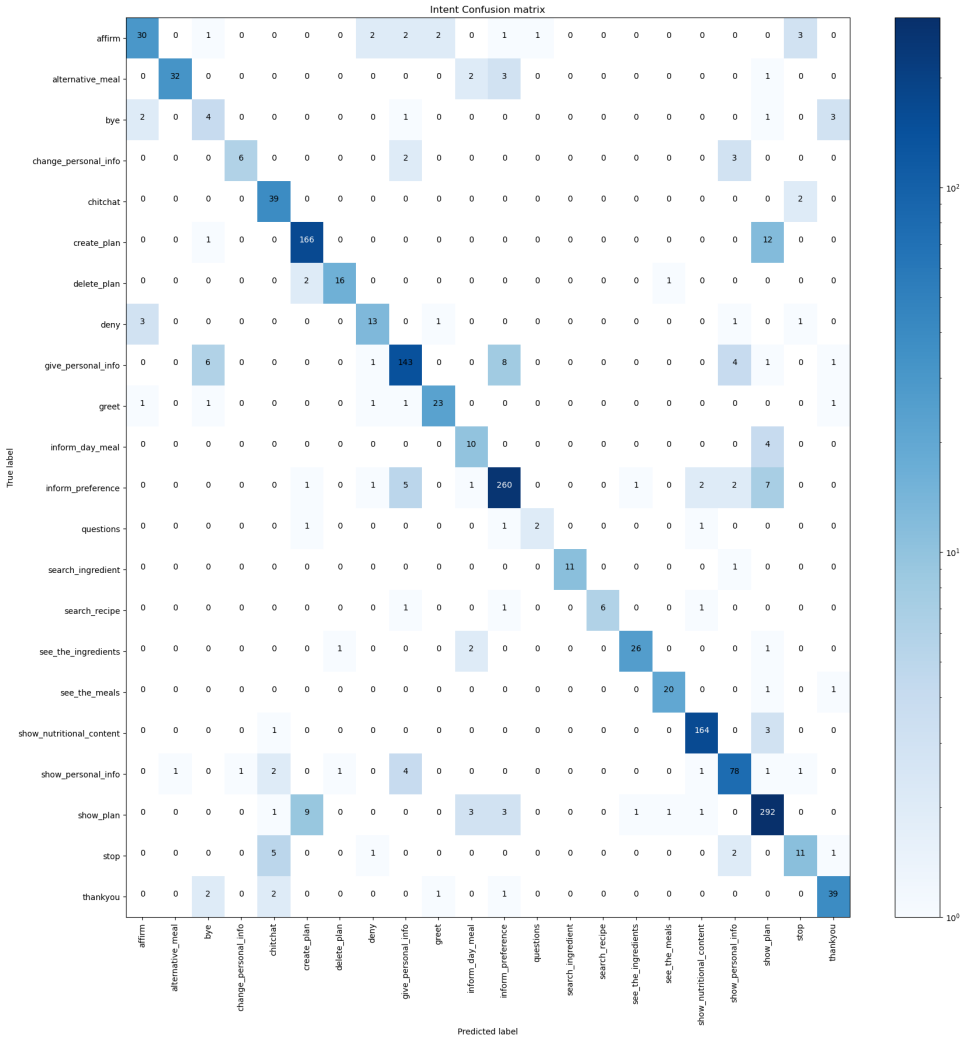


Figure A.4.1: Confusion matrix obtained for intent classification using the final NLU pipeline depicted in Section 5.1.8.

Appendix B

Complementary Information

B.1 Chatbot interface

The following Figures illustrate the interface used in the testing phase. Different actions are represented in these Figures.

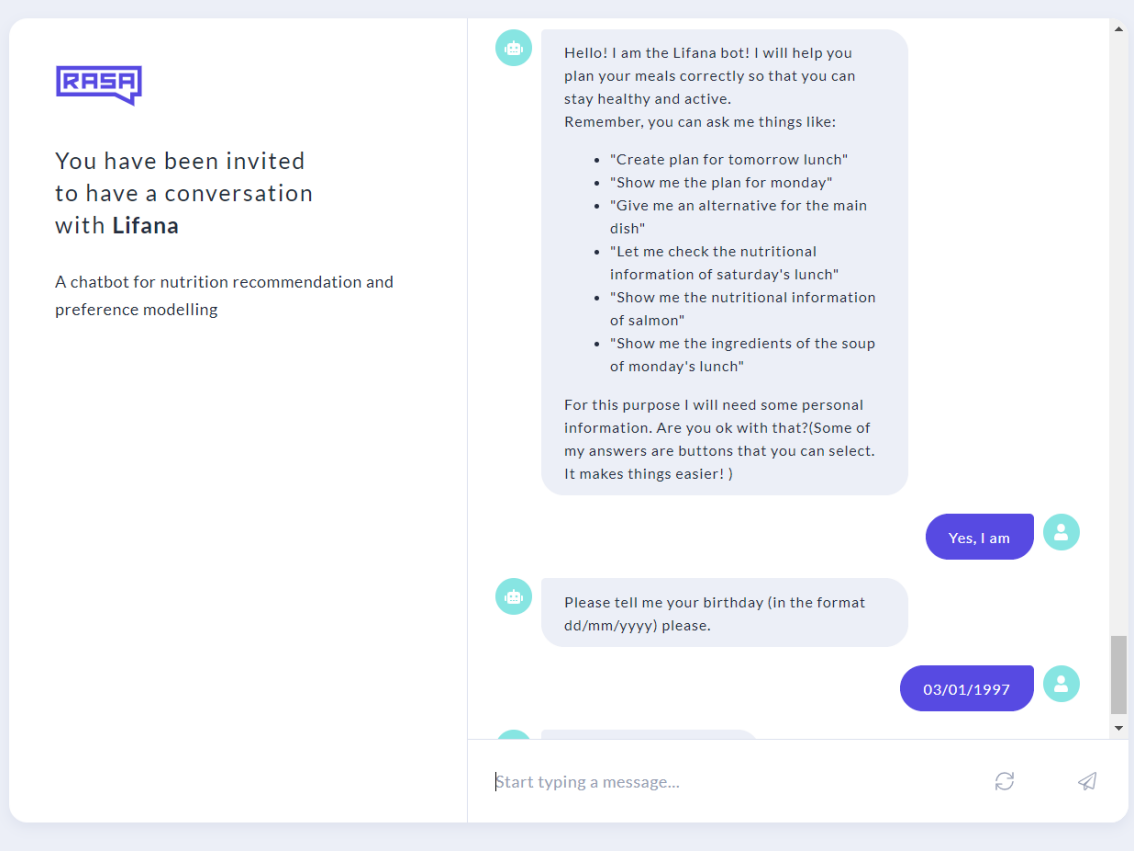


Figure B.1.1: Rasa X user interface used for user experience testing. This figure illustrates the initial message - before profile creation. It describes the scope of the chatbot and asks for personal detail collection

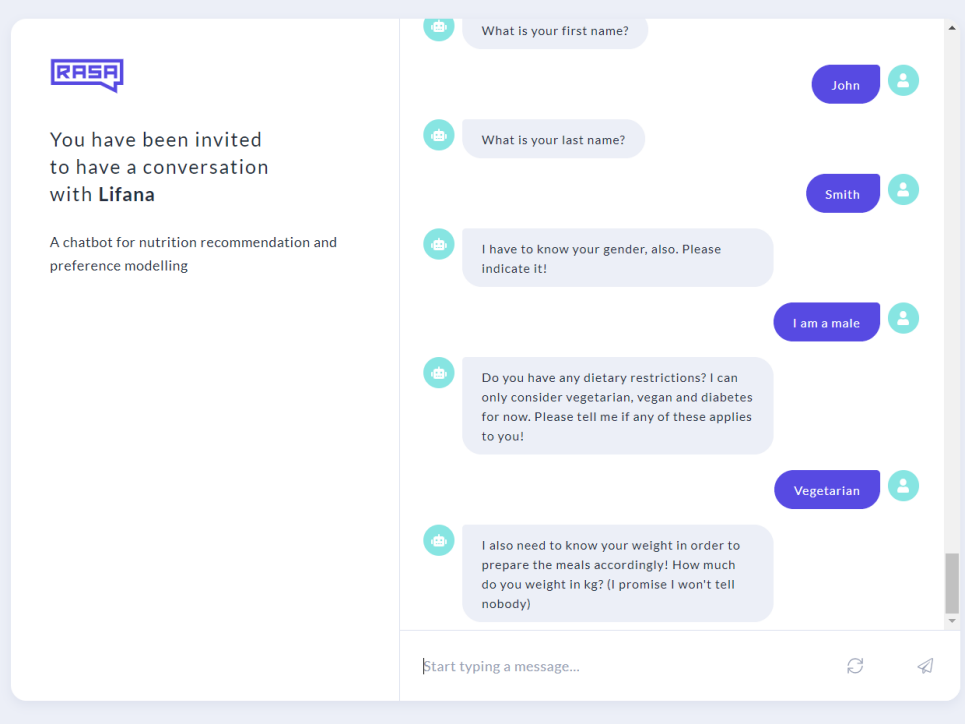


Figure B.1.2: Rasa X user interface used for user experience testing. This figure illustrates the further steps of personal detail collection

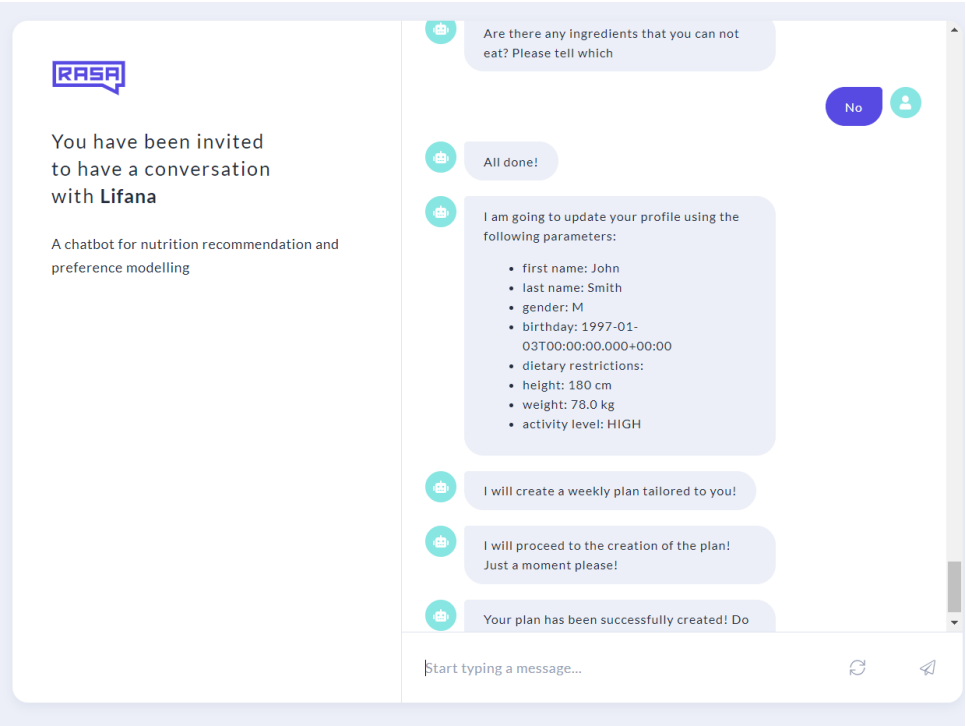


Figure B.1.3: Rasa X user interface used for user experience testing. This figure illustrates the last steps of profile creation

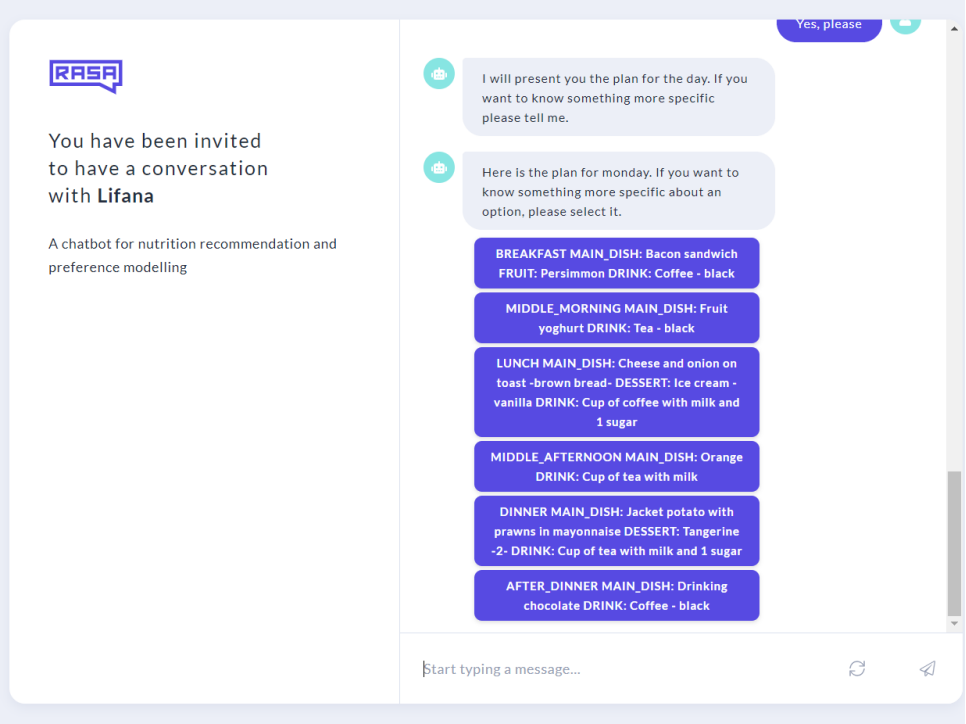


Figure B.1.4: Rasa X user interface used for user experience testing. This figure illustrates the process of requesting to see a plan. Also, the welcome message is customised according to the user's name.

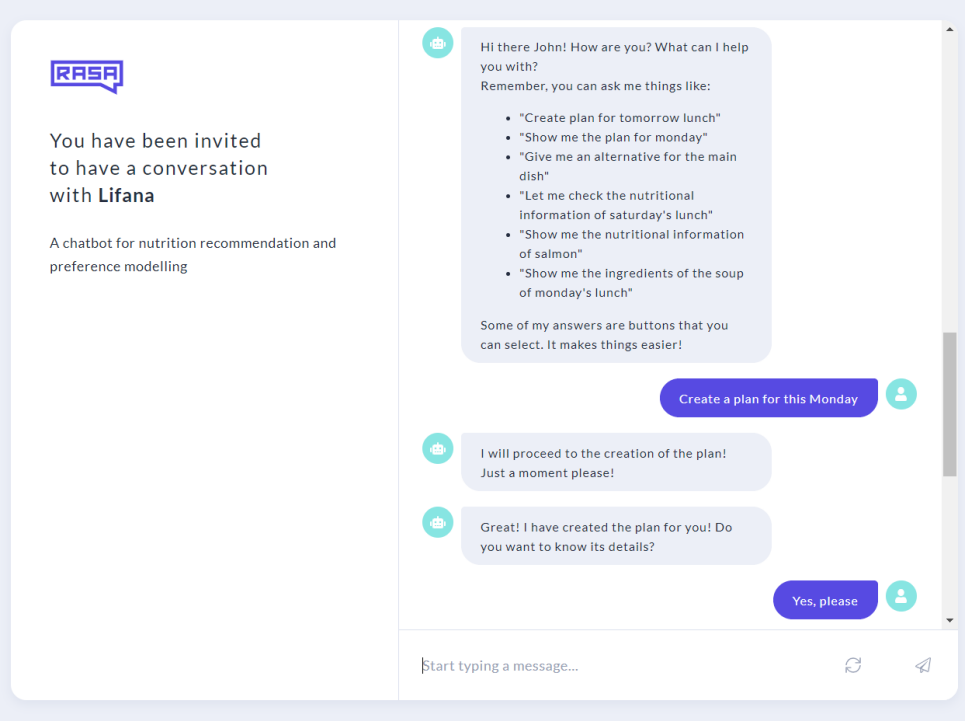


Figure B.1.5: Rasa X user interface used for user experience testing. This figure illustrates the process of plan creation

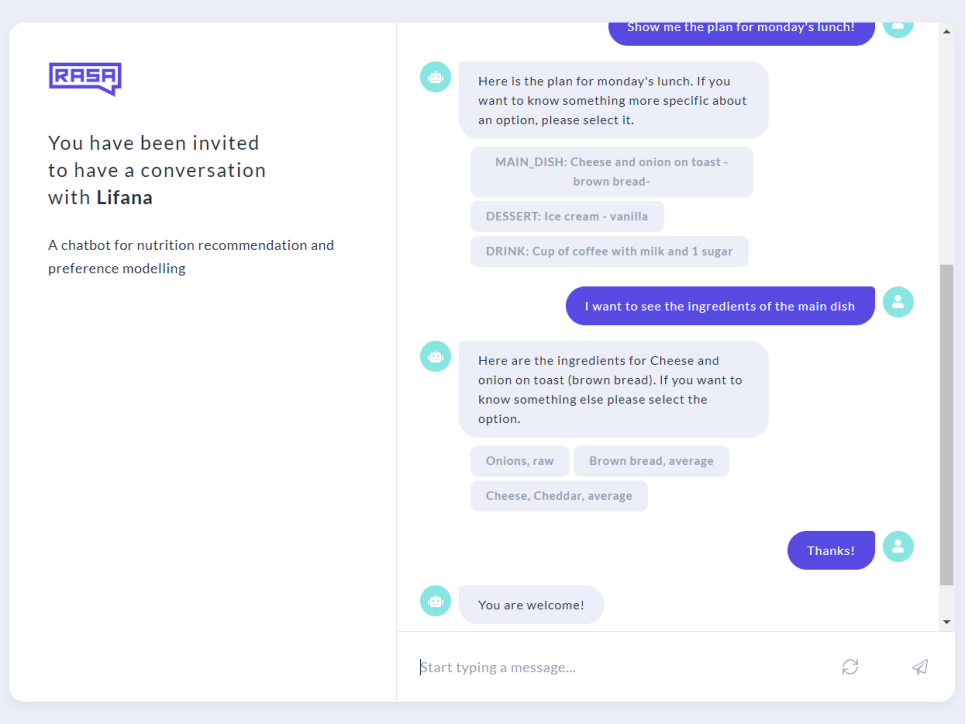


Figure B.1.6: Rasa X user interface used for user experience testing. This figure illustrates the ingredient consultation

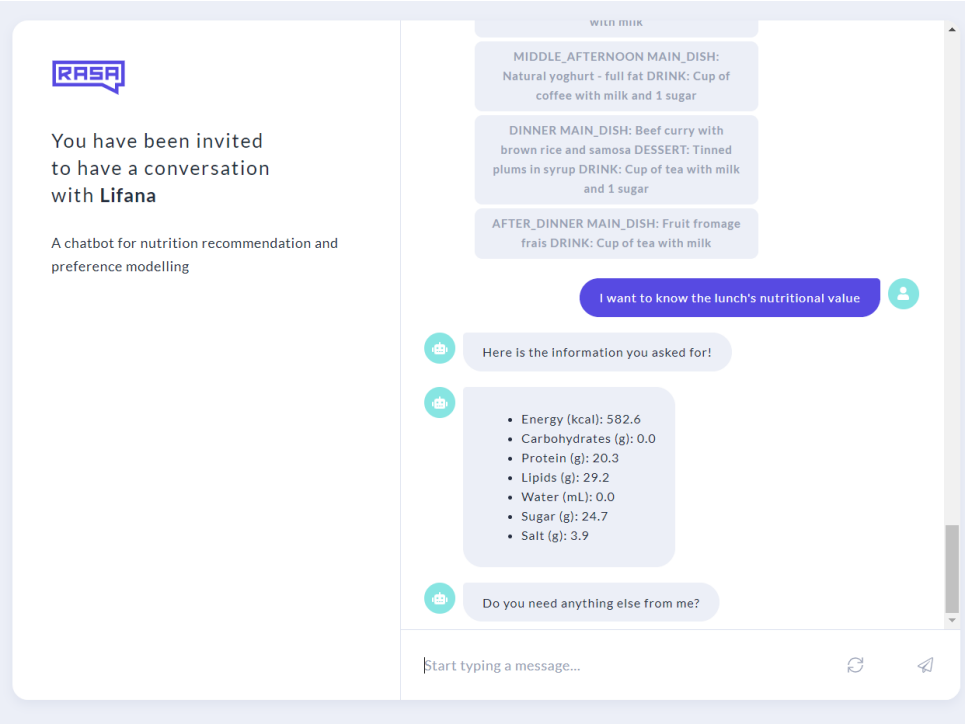


Figure B.1.7: Rasa X user interface used for user experience testing. This figure illustrates the process of requesting nutritional information

B.2 Evaluation Questionnaire – Google Form

This section includes both the protocol that was read at the beginning of each test session and the Google Form developed for User Evaluation. It is possible to observe the tasks and the several questionnaires.

B.2.1 Test Protocol

Under the scope of my Dissertation, a chatbot that is responsible for the interface between a nutritional recommender system and a user was developed.

This recommender system was developed by *Fraunhofer Portugal* with the help of a team of nutritionists and gathers information from several ingredient and recipe databases. Its focus is to create tailored weekly meal plans according to the user's nutritional needs. Currently, the interaction is made via mobile App, however a conversational interface is going to be tested in order to simplify interaction and increase engagement.

The following test session aims at adding a human judgement component to the chatbot's evaluation. Bearing this in mind, the test is divided in three parts: use cases tasks, food retrieval algorithm validation and usability and user experience questionnaires.

The first part consists of 5 tasks that must be completed by the volunteer and mimic the most common use cases of the system. Immediately before (pre-task) and after(post-task) each task a Single Ease Question must be answered. Also, the time of completion is going to be recorded for each task. Therefore, this test is ideally performed in a personal computer with the screen sharing option enabled. The tasks are explained in the Form that accompanies the test and all answers should be stored there.

An additional task is required in order to evaluate the food retrieval algorithm from the database, where two lists of ingredients will be displayed and 3 questions will be made to assess the correctness of the retrieval functions. The lists correspond to the response obtained by querying the database for a given ingredient using a word matching search and a semantic search.

After every task is completed, the volunteer must complete 3 questionnaires that evaluate the chatbot as a system from both a usability and a user experience perspectives (System Usability Questionnaire, User Experience Questionnaire and Chatbot Usability Questionnaire). The answers to these questionnaires should not consider the quality of the recommendations, since they are not part of the scope of this work, nor the Graphical User Interface used to talk to the chatbot as it is only an easy and convenient way to deploy the chatbot for user testing while having developer tools available. Moreover, the interface has a flaw where the messages sent by the volunteer only appear in the chat box after the answer from the chatbot. The aim of this evaluation is the conversational flow and the quality of the answers, as well as whether or not the chatbot provides the requested information correctly.

The link to the chatbot and the Form with the questions will be sent to the volunteers as they may proceed to fill out the demographic data part of the form and wait for further instructions.

Nutrition Chatbot Questionaire

Questions Responses

Section 1 of 17

Nutrition Chatbot Questionnaire

This questionare was developed to evaluate the user experience of the chatbot.

After section 1 Continue to next section

Section 2 of 17

Demographic data

Description (optional)

Name (optional)

Short answer text

App user *

- Yes
- No

Age *

- >15

- 18 - 23
- 24 - 34
- 35 - 44
- 45 - 54
- 55 - 64
- >65

Gender *

- Female
- Male
- Prefer not to say
- Other...

Schooling degree levels *

- High School (12°)
- Bachelor's degree (licenciatura)
- Master's degree
- Doctoral degree

How often do you use conversational agents/ smart assistants? *

- Never used
- Less then once a month

- Once a week
- Once a day
- Several times a day
- I really rely on them

After section 2 Continue to next section

Section 3 of 17

Tasks

In this section you are required to perform 5 different tasks. After each task you will be prompted with a

After section 3 Continue to next section

Section 4 of 17

First task (pre)

Description (optional)

Goal: Complete the chatbot's welcoming process, introduce your personal data

Description (optional)

Pre-task: On a scale of 1 to 7, how easy do you think it will be to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 4 Continue to next section

Section 5 of 17

First task (post)

Description (optional)

Post-task: On a scale of 1 to 7, how easy do you think it was to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 5 Continue to next section

Section 6 of 17

Second task (pre)

Description (optional)

Goal: Generate a meal plan for monday lunch and check the ingredients for the main dish

Description (optional)

Pre-task: On a scale of 1 to 7, how easy do you think it will be to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 6 Continue to next section

Second task (post)

Description (optional)

Post-task: On a scale of 1 to 7, how easy do you think it was to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 7 Continue to next section

Section 8 of 17

Third task (pre)

Description (optional)

Goal: Check the plan for monday dinner and express the preference towards one ingredient of it

Description (optional)

Pre-task: On a scale of 1 to 7, how easy do you think it will be to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 8 Continue to next section

Section 9 of 17

Description (optional)

Post-task: On a scale of 1 to 7, how easy do you think it was to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 9 Continue to next section

Section 10 of 17

Fourth task (pre)

Description (optional)

Goal: Create a plan for the next couple of days, check one of the day's dinner's dessert and

Description (optional)

Pre-task: On a scale of 1 to 7, how easy do you think it will be to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 10 Continue to next section

Section 11 of 17

Fourth task (post)

Post-task: On a scale of 1 to 7, how easy do you think it was to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 11 Continue to next section

Section 12 of 17

Fifth task (pre)

Description (optional)

Goal: Get the nutritional content for a day/meal/part/ingredient, after observing it

Description (optional)

Pre-task: On a scale of 1 to 7, how easy do you think it will be to complete this task? *

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 12 Continue to next section

Section 13 of 17

Fifth task (post)

Description (optional)

	1	2	3	4	5	6	7	
very hard	○	○	○	○	○	○	○	very easy

After section 13 Continue to next section

Section 14 of 17

Post test questions

The following sections consist of three usability and user experience questionnaires to assess the

After section 14 Continue to next section

Section 15 of 17

System Usability Score

Description (optional)

I think that I would like to use this system frequently *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I found the system unnecessarily complex. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

*

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I think that I would need the support of a technical person to be able to use this system. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I found the various functions in this system were well integrated. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I thought there was too much inconsistency in this system. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I would imagine that most people would learn to use this system very quickly. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I found the system very cumbersome to use. *

Strongly disagree Strongly agree

I felt very confident using the system. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I needed to learn a lot of things before I could get going with this system. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

After section 15 Continue to next section

Section 16 of 17

User Experience Questionnaire

Description (optional)

Question *

	1	2	3	4	5	6	7	
annoying (desagradável)	○	○	○	○	○	○	○	enjoyable (agradável)

Question *

not understandable (incompreensível)	○	○	○	○	○	○	○	understandable (compreensível)

Question *

	1	2	3	4	5	6	7	
creative (criativo)	○	○	○	○	○	○	○	dull (sem criatividade)

Question *

	1	2	3	4	5	6	7	
easy to learn (de fácil aprendizagem)	○	○	○	○	○	○	○	difficult to learn (de difícil aprendizagem)

Question *

	1	2	3	4	5	6	7	
valuable (valioso)	○	○	○	○	○	○	○	inferior (sem valor)

Question *

	1	2	3	4	5	6	7	
boring (aborrecido)	○	○	○	○	○	○	○	exciting (excitante)

Question *

not interesting (desinteressante) ○ ○ ○ ○ ○ ○ ○ interesting (interessante)

Question *

	1	2	3	4	5	6	7	
unpredictable (imprevisível)	○	○	○	○	○	○	○	predictable (previsível)

Question *

	1	2	3	4	5	6	7	
fast (rápido)	○	○	○	○	○	○	○	slow (lento)

Question *

	1	2	3	4	5	6	7	
inventive (original)	○	○	○	○	○	○	○	conventional (convencional)

Question *

	1	2	3	4	5	6	7	
obstructive (obstrutivo)	○	○	○	○	○	○	○	supportive (condutor)

Question *

	1	2	3	4	5	6	7	
good (bom)	○	○	○	○	○	○	○	bad (mau)

Question *

	1	2	3	4	5	6	7	
complicated (complicado)	○	○	○	○	○	○	○	easy (fácil)

Question *

	1	2	3	4	5	6	7	
unlikeable (desinteressante)	○	○	○	○	○	○	○	pleasing (atrativo)

Question *

	1	2	3	4	5	6	7	
usual (comum)	○	○	○	○	○	○	○	leading edge (vanguardista)

Question *

	1	2	3	4	5	6	7	
unpleasant (incómodo)	○	○	○	○	○	○	○	pleasant (cómodo)

	1	2	3	4	5	6	7	
secure (seguro)	○	○	○	○	○	○	○	not secure (inseguro)

Question *

	1	2	3	4	5	6	7	
motivating (motivante)	○	○	○	○	○	○	○	demotivating (desmotivante)

Question *

	1	2	3	4	5	6	7	
meets expectations (atinge as expectativas)	○	○	○	○	○	○	○	does not meet expectations (não atinge as expectativas)

Question *

	1	2	3	4	5	6	7	
inefficient (ineficiente)	○	○	○	○	○	○	○	efficient (eficiente)

Question *

	1	2	3	4	5	6	7	
clear (evidente)	○	○	○	○	○	○	○	confusing (confuso)

	1	2	3	4	5	6	7	
impractical (impraticável)	○	○	○	○	○	○	○	practical (prático)

Question *

	1	2	3	4	5	6	7	
organized (organizado)	○	○	○	○	○	○	○	cluttered (desorganizado)

Question *

	1	2	3	4	5	6	7	
attractive (atraente)	○	○	○	○	○	○	○	unattractive (feio)

Question *

	1	2	3	4	5	6	7	
friendly (simpático)	○	○	○	○	○	○	○	unfriendly (antipático)

Question *

	1	2	3	4	5	6	7	
conservative (conservador)	○	○	○	○	○	○	○	innovative (inovador)

After section 16

Continue to next section

Section 17 of 17

Chatbot Usability Questionaire

Description (optional)

The chatbot's personality was realistic and engaging *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot seemed too robotic *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot was welcoming during initial setup *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot seemed very unfriendly *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot gave no indication as to its purpose *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot was easy to navigate *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

It would be easy to get confused when using the chatbot *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot understood me well *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot failed to recognise a lot of my inputs *

1	2	3	4	5

Chatbot responses were useful, appropriate and informative *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Chatbot responses were not relevant *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot coped well with any errors or mistakes *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot seemed unable to handle any errors *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

The chatbot was very easy to use *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree