Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Medieninformatik

Conversational Information Retrieval for Instructional Content – A Modeling Framework and an Implementation for Recipe Search

Bachelor's Thesis

Jiani Qu

1. Referee: Prof. Dr. Benno Stein
2. Referee: Junior Prof. Dr. Florian Echtler

Submission date: August 6, 2018

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, August 6, 2018

. . . .
Jiani Qu

Abstract

As commercial voice agents are becoming increasingly popular, conversational search is brought into focus. Being an emerging area of research, there are a lot of questions that remain unanswered.

In this thesis, we aim to provide some insight into what a speech-based conversational search system might look like, focusing on its application in recipe search, as cooking is a typical hands-off situation where voice interactions and information retrieval with an agent afford more convenience. Our contribution is two-fold: (1) a human-like conversational information retrieval model, and (2) a prototype of conversational recipe search.

Contents

1 Introduction 1

2 Background and Related Work 4
  2.1 Conversational Search 4
  2.2 Alexa Skills Development 6
  2.3 Related Work 7

3 Conversational Information Retrieval Model 9
  3.1 Conversational Search Model 9
    3.1.1 Search Query Formulation 10
    3.1.2 Search Query Refinement 10
    3.1.3 Search Result Exploration 10
    3.1.4 Search Result Confirmation 12
  3.2 Conversational Result Interaction Model 12
    3.2.1 Single-Turn Interaction 13
    3.2.2 Multi-Turn Navigatable Interaction 13

4 Corpus Construction and Statistics 15
  4.1 Crawling of Online Portals 15
  4.2 Mining and Corpus Document Structure 17
  4.3 Indexing of Recipe Documents 18
  4.4 Statistics 19

5 Conversational Recipe Search Prototype 22
  5.1 Recipe Search 24
    5.1.1 Retrieval 26
    5.1.2 Ranking 27
    5.1.3 Single-Turn Q&A 27
  5.2 Cooking Instruction 27
    5.2.1 Similarity Computation for Information Division Retrieval 28
    5.2.2 Machine Comprehension for Q&A 30
  5.3 Conversational Search Properties of the Prototype 31

5.4 Interaction Approaches of the Prototype 32

6 Conclusion and Future work 34
6.1 Contributions 34
6.2 Future Work 35
6.2.1 Evaluation Study Proposal 35
6.2.2 Other Future Work 37

Bibliography 39

Chapter 1

Introduction

Using our voice to access information and interact with an intelligent system has long been part of science fiction. Due to recent advances in automatic speech recognition (ASR) and text to speech synthesis technologies, commercial chat bots and voice assistants are becoming increasingly popular, bringing science fiction scenes into reality.

In 2016, Google reported that approximately 20 percent of mobile queries are submitted via voice inputs [1]. In addition, virtual assistants such as Apple Siri, Google Assistant, Amazon Alexa and Microsoft Cortana not only allow users to find information, but are also slowly integrating into everyday life for tasks from setting up a timer to booking tickets for a film. Voice-only interaction provides users with the possibility of hands-off convenience, and is for example preferred, when operating machinery, when no screen or keyboard is available, when users are on the move [Trippas, 2016], or when they are cooking. That is also why our research focuses on conversational search in the context of cooking.

As an emerging research direction in Information Retrieval (IR), the conversational approach to IR was identified as one of the most important directions by a report from SWIRL 2012 [Allan et al., 2012]. It was also indicated that there is a lack of understanding of search tasks, search result description, search application and evaluation of speech-based conversational systems [Joho et al., 2017]. Moreover, replicating the classic approach is ineffective over a speech-only channel [Sahib et al., 2012b]

In a classic Information Retrieval setting, the system answers to a single natural language request describing results the user wants (query) based on

---

[1] `https://www.youtube.com/watch?v=862r3XS2YB0`

a database where information is stored to allow for quick retrieval (index). Previous interactions in the same session can be stored and learned by an advanced system to provide more desirable results within the session [Eickhoff et al., 2014]. However, in natural language conversations, people expect additionally that information contained in previous interactions can also be referred to, even implicitly [Kenter and de Rijke, 2017] — in another word, contextual meaning should be well preserved, which most of the systems still struggle with.

Apart from retaining contextual information, it was put forward by Radlinski and Craswell [2017] that a conversational search system should be able to build a cumulative picture of the user's information need based on their query statements and other relevance feedback over time.

We briefly assessed the ability to find recipes and instruct cooking from two of the aforementioned commercial voice assistants in speech-only scenarios. Backed by a powerful search engine, Google Assistant can find recipes with the given name or ingredients. However, it is, in its core, still the standard search approach, with on-click retrieval. Alexa does not support recipe search with its built-in functions, but with third-party applications. Some of them perform well in searching, but still lacking in allowing relevance feedback, for example, users not being able to adjust ill-formed queries.

Although the topic of conversational search came into view already decades ago, it was not until recent years that it came into focus. Radlinski and Craswell [2017] are the first that formally defined the term *"conversation"* from an IR perspective, suggested a set of required properties that can be used to measure the conversational extent of a system, and designed a theoretical framework. As no truly intelligent conversational search systems exist yet, other relevant research mainly explores user requirements based on user studies, of which we'll discuss more in Chapter 2. Taken all together, a lot of questions still remain unanswered in the field, such as:

- How should a new information seeking model for a speech-based conversational system look like?
- What are effective techniques to present results using audio?
- How to evaluate the effectiveness of a conversational system?
- How to understand users' information need in a dialogue?

We aim to find answers of the first three questions.

In this thesis, we contribute with the first computational implementation and evaluation proposal with the theoretical framework in recipe search and cooking assistant context. We present a conversational information retrieval model and a prototype cooking assistant that is able to do recipe search and instruct cooking as an Amazon Alexa skill. With that, we also advance the knowledge on voice-based conversational search in (i) answer seeking strategy, (ii) user information need, and (iii) result presentation.

The model is split into two parts: search and "within-document" retrieval (interactive presentation of retrieved result document's information). Details of the model and its correlation to the theoretical framework will be explained in Chapter 3. To back the search, we crawled and indexed a total of 17,308 recipes from wikiHow and eHow, exploiting the sites' structure to avoid mining and annotation errors while unifying the recipes based on the model. The corpus construction and its statistics are contained in Chapter 4.

With the interaction model and indexes, we built the cooking assistant prototype involving many techniques such as machine comprehension and others that are discussed and documented in Chapter 5. Furthermore, in the same section, we analyze to what extent our prototype satisfies the desired properties of conversational search. Finally, this thesis is concluded in Chapter 6, where we summarize our findings, propose an evaluation study of our prototype and discuss other future work.

Chapter 2

Background and Related Work

In this chapter, we introduce the necessary background for this thesis. First, in Section 2.1, we start with describing the work undertaken by Radlinski and Craswell [2017] to define conversational search and its properties, and conclude a general theoretical framework, which we take as our research context. Next, the Amazon Alexa Skills Kit is introduced in Section 2.2, with which our prototype is developed, so that an understanding of the advantages and restrictions can be reached in later chapters. Last but not least, we give an overview of previous efforts made on providing insight into spoken conversational systems in Section 2.3.

2.1 Conversational Search

With the recent advances in machine learning and ASR, speech-based conversational search came into focus. Radlinski and Craswell [2017] define a conversational search system as "a system for retrieving information that permits a mixed-initiative back and forth between a user and agent, where the agent's actions are chosen in response to a model of current user needs within the current conversation, using both short- and long-term knowledge of the user." According to them, the system has the following five properties:

User Revealment: Helping users express or even discover their true information need, and possibly also long-term preferences

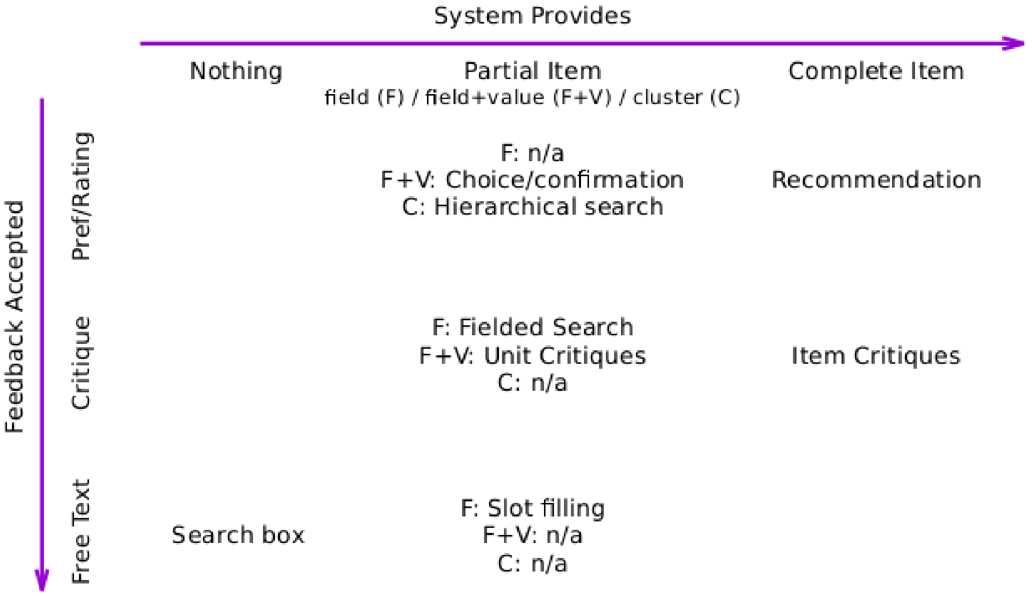
System Revealment: Informing users about its capabilities to build users' expectations

Mixed Initiative: The system and user both being able to take initiative as appropriate

Memory: User being able to reference past statements and stances

Set Retrieval: Ability to reason about the utility of sets of complementary items.

They suggested a theoretical conversational search model that summarizes existing conversation settings into nine interaction approaches as shown in Figure 2.1. They are also described as *action spaces* of the system, for the system provides three basic types of information and expects three types of feedback, with provided information falling into: nothing, a partially described item, and a completely described or specific item. For example, "a German cake" is a partial item, and "black forest cake" is a complete item. Particularly, an item may be partially described in multiple ways, from the simplest case of selecting a feature (field), to adding extra value of each field such as "cooking time under 30 minutes", and finally presenting a cluster of items that can be grouped together according to similar features like "pancakes based on yogurt instead of milk".



	Nothing	Partial Item field (F) / field+value (F+V) / cluster (C)	Complete Item
Pref/Rating		F: n/a F+V: Choice/confirmation C: Hierarchical search	Recommendation
Critique		F: Fielded Search F+V: Unit Critiques C: n/a	Item Critiques
Free Text	Search box	F: Slot filling F+V: n/a C: n/a	

Figure 2.1: Conversation action space, as matched to previous names from past work. The system may provide three types of feedback, and expect three types of responses in return. [Radlinski and Craswell, 2017]

What information users are expected to provide for the search can also be categorized vice versa. With increasing complexity, these are: simple preference between choices or preference score in response to a question; providing reasons for how the presented item fails to meet the user's information need; and free text, meaning that the user may specify any possible feedback.

2.2 Alexa Skills Development

Amazon Alexa provides skills, or capabilities that can also be understood as voice applications, that allow users to create a more personalized experience [1]. Our conversational recipe search prototype stands as a Alexa Skill developed with the Alexa Skills Kit.

Each skill is based on a collection of user intents, sample utterances, and the dialogue model [2], which is named as an Interaction Model – we will address this as Alexa Interaction Model (AIM) in the remainder of this thesis to avoid confusion with our interaction model. **Intents** are user requests the skill can handle; **sample utterances** are the words and phrases users can say that are mapped to the intents; a **dialog model** identifies information the skill requires and the prompts Alexa can use to collect and confirm that information in a conversation with the user. In addition, an intent can have optional **slots** that represent certain sets of words.

However, except the dialog model, with which developers define required slots, the prompt from Alexa if slot values are missing and if Alexa should ask users to confirm slots or the entire intent, the AIM itself contains no logic between user request and agent response.

To develop a custom skill, we have to define an invocation name with which the user opens the skill, for example in our case "Open recipe search". Then, we declare intents and their sample utterances. Alexa learns the utterances, so that users' spoken requests can be recognized and mapped into corresponding intents even when they are not exactly the same as the samples.

Slots can be embedded in sample utterances in cases where users are expected to say words or phrases that belong to the same category, for example flight destination, debate topics, food genres, or simply prepositions etc. This also has to be learned, so a set of example words have to be established. Unlike sample utterances which we don't have access to on the back-end, slot words

---

[1] `https://developer.amazon.com/alexa-skills-kit`

[2] `https://developer.amazon.com/docs/custom-skills/steps-to-build-a-custom-skill.html`

or phrases can be read and further processed.

With the support of AIM, developers further determine and program the responses Alexa gives to different user intents and host the program as an endpoint server, ready for whenever a user tries to interact with the skill.

2.3 Related Work

Already over two decades ago, Belkin et al. considered a conversational approach to information retrieval by characterizing information-seeking strategies. They reintroduced a dialogue structure, the Conversational Roles Model (COR) for the situation of information seeking (Figure 2.2) that tends to hold for the genre of cooperative information-seeking dialogues in a human-computer interaction environment [Stein and Thiel, 1993]. The COR represents a recursive network as a basic schema of a dialogue and includes interrelations of dialogue acts, e.g. asking, offering, answering, etc.

Belkin et al. also proposed the concept of *"script"*, which is a prototypical interaction pattern or plan for a dialogue between the user and the system [Belkin et al., 1995]. The system follows the scripts using case-based reasoning to select next steps and offer users choices, forming a mix-initiative dialogue which is often a characteristic of conversations [Walker and Whittaker, 1990].

In another study, by comparing the user experiences on information search with three different agents, namely, a commercial system (Google Assistant), a human expert and a human disguised as an automatic system (wizard), Vtyurina et al. [2017] discovered some of the user expectations of voice search that further indicate functionalities such as context preservation, feedback allowance, information source provision and information need detection will improve user experience.

In addition, as one of the first to research on the topic, Trippas et al. [2018] carried out a first major study to provide insight into what a spoken conversational system may look like. They examined the information-seeking process in-depth at three stages: query formulation, search result exploration and query reformulation [Sahib et al., 2012a] in a human-human setting. The study suggests that a spoken system has increased complexity and interactivity in that, users, not being confined to a search box anymore, often specify multiple moves/requests in one utterance (Figure 2.3) and the idea of static boundaries between the standard search engine result page and its documents appears to fade. With the observations, it was reported that existing search behavior models do not fit spoken conversational systems due to their lack of interaction or flexibility thereof, including the aforementioned COR model of Stein and Thiel and Belkin et al.'s scripts.

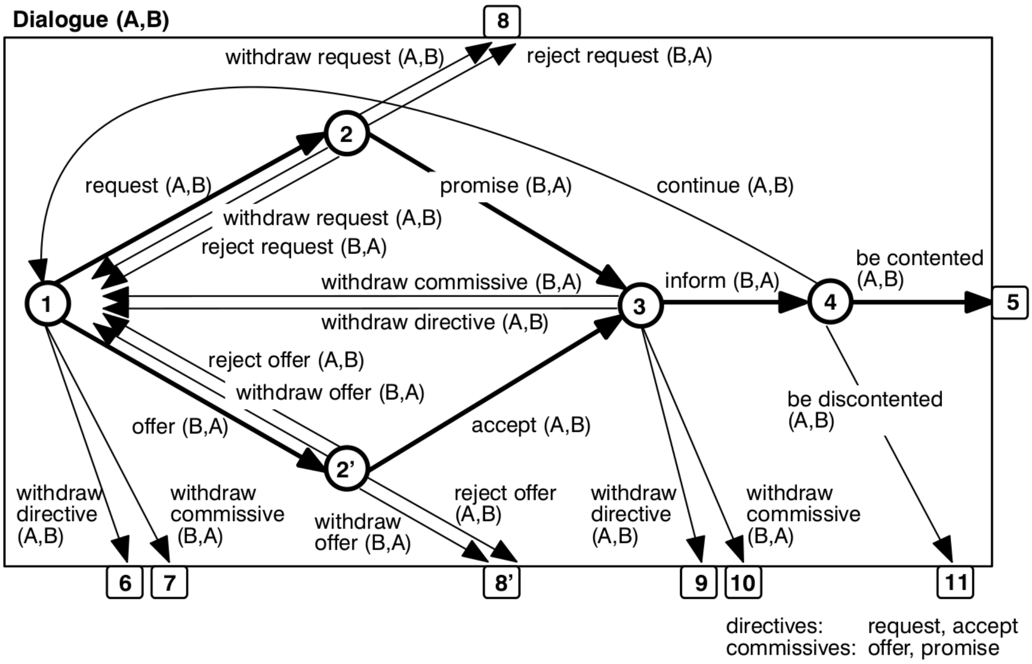
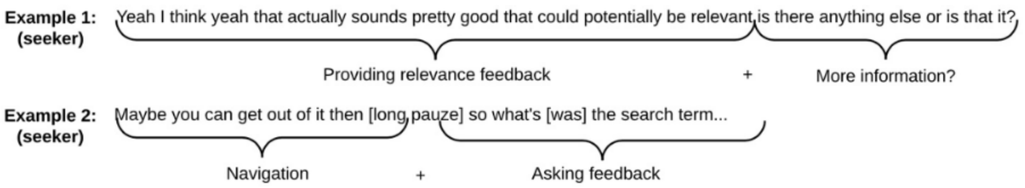


Figure 2.2: Network representing the basic COR "dialogue" schema. Circles are states within a dialogue, squares indicate terminal states. A refers to the information seeker, B to the information provider. The first parameter indicates the speaker, the second the addressee. [Stein and Thiel, 1993]


Example 1: (seeker) Yeah I think yeah that actually sounds pretty good that could potentially be relevant is there anything else or is that it?

Example 2: (seeker) Maybe you can get out of it then [long pauze] so what's [was] the search term...

Figure 2.3: Multiple moves examples [Trippas et al., 2018]

Chapter 3

Conversational Information Retrieval Model

In the last section (Section 2.3), we mentioned that Trippas et al. reported that existing search behavior models do not fit spoken conversational system due to their lack of interaction or flexibility. We aim to develop a new model that contains the required flexibility and is broadly applicable without domain specific restrictions.

The interaction model we present here serves two Information Retrieval purposes: (1) document search, and (2) result interaction. For this model, we assume that there is a user searching for information and a speech system is assisting the user.

Moreover, we divide the result interaction that involves in-document retrieval further into two categories: the Q&A case, and the multi-turn navigatable case. For the latter one, we presume the user is searching for a large piece of information that would result in the user's cognitive overload if read without further interaction between the user and system.

In Section 3.1, we introduce our conversational search model and in Section 3.2, the result interaction model will be explained in detail.

3.1 Conversational Search Model

We define the interaction between the user and the system mainly in four stages: (1) Search Query Formulation, (2) Search Query Refinement, (3) Search Result Exploration, and (4) Search Result Confirmation.

We also classify interaction intents between the user and agent into 8 classes shown in Table 3.1. Six of the classes (OQ, IR, FD, PA, NF, PF) were introduced by Qu et al. [2018]. We added two more: *Intent Correction* and *Navigating Directive*. The original set of classes represents only the text-based online dialogue in a human-human situation, we added IC that is prominent during a human-machine conversation, since an intelligent agent depends on user audio quality to understand a query, and might thus interpret a phrase or word wrongly where the user should be given the ability to correct the machine. Other than that, we redefined *Information Request*, so that it includes all questions apart from the original query.

3.1.1 Search Query Formulation

This is the starting point for most Information Retrieval systems where the user can formulate any question. It is equivalent to the Null System – Free Text User case that Radlinski and Craswell [2017] introduced in their theoretical framework for conversational search. In our model, Query Formulation always represents the start of a new search that is signaled by an original query.

3.1.2 Search Query Refinement

As mentioned prior, we take it into consideration that the user may correct the query at any time if the agent fails to transfer audio into text precisely. Also, when the agent system is presented with a query that is not concrete enough or in order to limit the number of results, the system may take the initiative to request more information or clarification from the user to further filter possible returns; alternatively, when a user is dissatisfied with a given result, he or she may also reformulate the query for example by adding or removing search parameters.

3.1.3 Search Result Exploration

This is the stage where users explore returned results. On a conventional web search engine, a list of results are presented graphically all at once. However, conveying a large amount of information via speech would very possibly lead to the user's cognitive overload [Lai and Yankelovich, 2006] [Melto et al., 2008], as already pointed out by Trippas et al. [2015].

To allow users to explore results conversationally, the system provides a restricted amount of options per turn that, put together, does not exceed a certain length, and waits for the user's reaction or feedback. To find a desirable

Table 3.1: Descriptions and examples of intent classes.

Code	Label	Description *Example*
OQ	Original Question	The first question by a user that initiates the dialog with an agent. *Which meat dishes are there?*
IR	Information Request	Users or agents ask for more information on something. *Is this recipe vegetarian?*
FD	Further Details	Users or agents provide more details. *I prefer chicken meat over the others.*
IC	Intent Correction	Users correct a misunderstood intent. *No, I mean a cake that doesn't need baking.*
PA	Potential Answer	A potential answer or solution provided by agents. *I found this recipe: ...*
NF	Negative Feedback	Agents provide negative feedback when nothing can be found; or users giving negative feedback on undesired results. *No that's not what I'm looking for.*
PF	Positive Feedback	Users provide positive feedback on the result. *Yes, that's what I'm looking for.*
NG	Navigating Directive	Users order agents to go back, forth, pause, repeat etc. *Go back to the last step.*

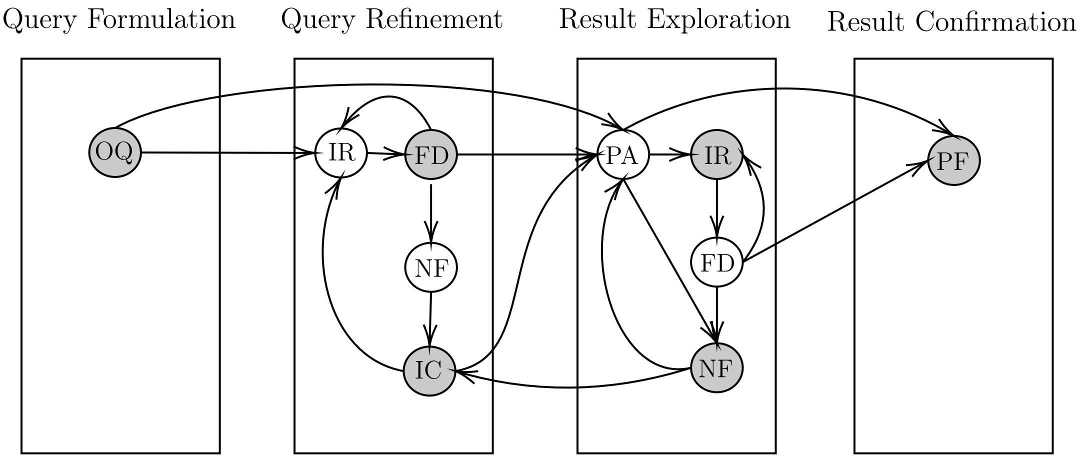


Figure 3.1: An example intent flow in our conversational search model. Gray node: users. White node: agents.

item, the user may request further information from the returned documents and afterwards decide if a next turn should follow. This could involve machine comprehension of the document text, which is also a form of result interaction that we'll discuss in more detail in our implementation of a prototype in Section 5.2.2.

3.1.4 Search Result Confirmation

Once the user gives a positive feedback on a certain result, it signals the start of result interaction in form of document presentation. It is notable that this does not mean the termination of the search process as the user can go back to explore results at any time inside a session.

An example intent flow in our model is shown above in Figure 3.1. As can be seen, it is possible for the agent to immediately give a Potential Answer after the user's Original Question, meaning, the Search Query Refinement phase is not mandatory. The Information Request from the user's side during Result Exploration might be an information need in the found document's content, where the agent will proceed shortly with result interaction. Result Confirmation, like said before, is also the signal of result interaction. Users can return to the search phase at any time, whether to find a more desirable result or to interact with multiple results at the same time if needed (e.g. comparing parameters of two computers).

3.2 Conversational Result Interaction Model

The conversational result interaction is very different from the index-based search. The unit of results that index-based search returns is the document. The conversational result interaction however, either serves the purpose of finding short answers to the user's questions about the document content, or returns the needed information in multiple turns if the information load is too huge. In both cases, the unit of returned results is text snippet extracted from the document.

To illustrate this mathematically, we assume a document is composed of atomic information units. We define all the information contained in the document to be a set $\mathcal{C}$ as a collection of information units u_i, $u_i \in \mathcal{C}$, $i > 0$. The answer $\mathcal{A}$ according to a user's information need can be of various length in different cases. If (1) the length does not exceed a certain threshold that can cause the user's cognitive overload, $\mathcal{A}$ will be returned directly in a single turn, $\mathcal{A} \subseteq \mathcal{C}$; otherwise if (2) the length exceeds the threshold, $\mathcal{A}$ has to be split into n parts: $\mathcal{A} = \bigcap_{j=1}^{n} \mathcal{A}_j$, where the length of $\mathcal{A}_j$ is below the threshold and $u_i \in \mathcal{A}_j \subset$, $0 \leq j < n$. For example, $\mathcal{A}$ is the step-by-step instruction of a recipe, $\mathbb{A}_i$ is a step that contains several sentences.

We describe case (1) as the single-turn interaction and case (2) multi-turn navigatable interaction.

3.2.1 Single-Turn Interaction

As just mentioned, if the user required information is within a certain length that won't result in the user's cognitive overload, the agent will return the answer directly in one turn. This corresponds with a simple single-turn Q&A model.

According to Kirsh [2000], a cause of cognitive overload that applies here is the oversupply of pushed information over which the users have only short term control. One thing that needs to be noted is, because the information needs in different scenarios vary from each other, the amount of information that reaches the state of oversupply is also different. For example, if the users ask for news reports, they expect a large amount of information input in a short amount of time; however, if they are multi-tasking, say, handcrafting and need instruction, their task already takes up space in their limited working memory [Miller, 1956], so only information relevant to their actions will be expected. The former case is where the single-turn result interaction is applicable; while for the later, the multi-turn navigatable interaction is more appropriate, which we'll discuss next.

3.2.2 Multi-Turn Navigatable Interaction

On a graphic interface, the text can be re-read, which means that most people absorb written information better than spoken information [Yankelovich et al., 1995]. To assure that users' cognitive overload can be reduced or eliminated, they can feel more in control of their tasks and that the information can be fully utilized, we define the multi-turn navigatable interaction in two phases: (1) Document Overview and (2) Information Unit Exploration.

Document Overview

In order to utilize the information contained in an answer or a result, users first need to have an idea of how much or what kind of information is present and how it can be accessed. In this phase, agents provide users with an overview of the available information divisions of the result, e.g. in a handcraft scenario: tools, tips and parts of the procedure; in an argument search: different conclusions, topic shifts etc.; in a long text scenario: the summary of each paragraph. Agents might also inform users about which prompts they can make or what the system is able to do.

Information Unit Exploration

This is the stage where users navigate through information units or information unit collections contained in the results. It starts with the user choosing a browsing destination, namely a division that is listed in the overview or a division that is not listed, but the agent is able to extract from the whole document. Since the division could be a large chunk of information, the agent returns an information unit or a small set of information units each turn, with automatic pauses in between turns or on the user's cue (Navigating Directive). The control is completely in the users' hands in that they can navigate through the result freely, which includes going back, going forth, repeating, jumping to a certain point etc., so that their information need can be better fulfilled.

An example intent flow of the conversational result interaction is illustrated below in Figure 3.2. The overview of the document provided by agents is categorized in Further Details, so is the return of information units or clusters. During information unit exploration, users are also able to make information requests at any time as shown in the figure. And as mentioned in the last section, users can also return to the search phase whenever they will.

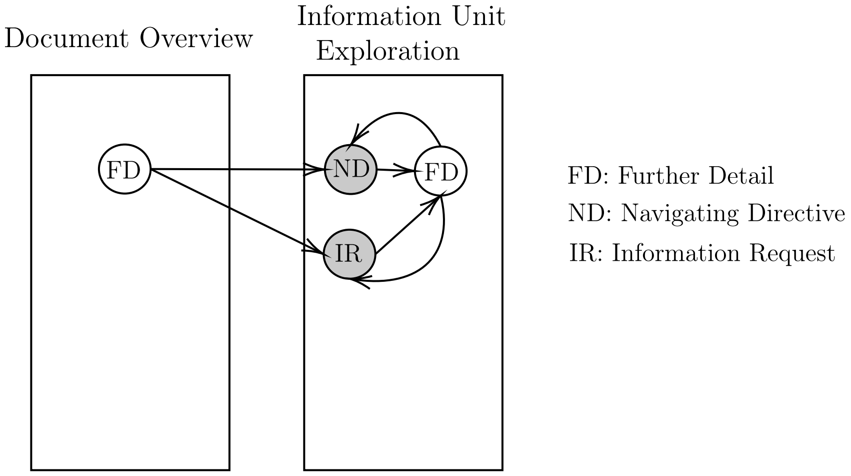


Figure 3.2: An example intent flow in the conversational result interaction model. Gray node: users. White node: agents. See Table 3.1 for intent classes.

Chapter 4

Corpus Construction and Statistics

Our model previously introduced in Chapter 3 requires a combination of search index and data collection. In order to build a conversational recipe search prototype that implements our proposed model, we created an initial data basis.

In this chapter, we provide insights into the corpus creation process and statistics of the data. Our data collection process is carried out in three steps: raw-data collection, mining and corpus construction, and indexing, which will be explained in detail in Section 4.1, Section 4.2 and Section 4.3 accordingly. In Section 4.4, we present statistics of our corpus and index.

4.1 Crawling of Online Portals

We collected data from two online how-to guides portals, *eHow* and *wikiHow*. They present a large database of step-by-step instruction articles covering a wide variety of topics from food and drinks to crafts and repair.

We crawled articles of category "Food and Entertainment/Recipes" from wikiHow and that of "Food & Drinks" from eHow. For the former portal, we used its site-map to access all published articles with simple HTTP GET requests as the site allows crawling real contents that are also static. Since the article URLs on the site-map are not categorized, we first crawled all articles available and then filtered out desired ones by their category information.

For eHow, because the site disallowed robots, a site-map is not available and all contents are dynamic, we had to crawl page sources using a webbrowser virtual machine with the help of selenium [1]. However, many articles under "Food & Drinks" are food preparation techniques, food facts and etc, but unlike wikiHow, neither the given categorization of the articles nor page meta-data allows us to filter out the recipes. This resulted in food-facts also being present in our corpus. Altogether, we end up with a raw data collection of 9,751 and 8,302 HTML files of wikiHow and eHow respectively.

Table 4.1: Overview of the article structures from wikiHow and eHow. "P" stands for paragraph. The separated lower parts are instruction-relevant elements. If an element is mandatory, it is always present.

Portal	Element	Subelement	Subsubelement	Mandatory
wikiHow	Title			✓
	Introduction			X
	Ingredients			X
		Part-Ingredients		X
	Things You'll Need (tools)			X
	Quick Summary			X
	Category Hierarchy			✓
	Related Articles			✓
	Tips			X
	Warnings			X
	Steps/Methods/Parts			✓
		Step		✓
			Step description	✓
			Step details	X
			Step specific tips	X
eHow	Title			✓
	Introduction			✓
	Things You'll Need (ingredients AND tools)			X
	Category Hierarchy			✓
	Related Articles			✓
	Tips			X
	Warnings			X
	Sections/Body text/Steps			✓
		Section/P/Step		✓
			Description	X
			P/Sentence/Step details	✓
			Specific tips	X

---

[1] `https://www.seleniumhq.org/`

4.2 Mining and Corpus Document Structure

WikiHow, being open-source and open-content where all users can publish their own content or edit existing ones on the site, both the recipe articles' structures and HTML structures are surprisingly well-kept and uniform, which simplified our information extraction.

Yet, on eHow, despite the fact that its articles are only created by its employed writers, both the article structures and HTML-element tags differ widely from each other. While the most recent posts largely conform with the clear step-by-step structure of wikiHow, many older posts present themselves in other ways. We mainly discovered 3 most representative structures: (1) recipes in small steps similar to that of wikiHow, (2) articles as whole text without any sectioning, and (3) articles in sections with each section as whole text. Moreover, their HTML-element tags and names are also different, which produced obstacles for our mining process. We summarized the article structures of both portals in Table 4.1.

According to Table 4.1, it is not difficult to observe, that generally, the structures are still similar. One thing worth noting is, many articles contain more than one recipe, that are specified under different methods, parts or sections. However, the ingredients for each are not necessarily segmented accordingly. Thus, we extract exact one JSON document from each valid article.

The resulting corpus document has the following construction:

```
title:string,
articleID:string,
intro:string,
category:[string],
related:[string],
tips-warnings:[string],
ingredients-thingsneeded:[{
      part-ingredients:[string],
      part-name:string}],
parts:[{
      part:string,
      steps:[{
            step:string,
            sub-steps:[sub-step:string],
            tips:[string]}]}]
```

All ingredients and tools needed will be stored together in "ingredients-thingsneeded", with each specified list stored alone as one JSONObject. "parts" contains different sections or different methods of an article. If the article is without sectioning, the whole is considered as one section. If the title of a section is present, it is stored as "part".

"Step" and "sub-step" represent two levels of abstraction of a cooking process. The high level "step" usually gives a short summary, such as "Serve the roast eel"; whilst "sub-step" explains the cooking process in detail, for example "Top the eel with lemon, salt and pepper, and any other sauce or garnish that you enjoy."

Given the previously mentioned 3 representative article structures, section content is split into steps according to (1) steps specified in article, (2)&(3) paragraphs. For case (2)&(3), because a summary abstraction is not given, the whole paragraph is stored as "step", while "sub-step" is left empty – albeit this current approach suffices for our first prototype, step sectioning from a long paragraph can be improved in the future with machine learning methods.

Not all articles that have multiple methods specify correspondingly multiple ingredient lists. Many have only one list that contains all ingredients, and some others have unequal numbers of methods than ingredient lists. This is why we did not store ingredients according to methods, but separately, which results in a twist in retrieval that will be explained later in Section 5.2 in the prototype chapter. We can, however, imagine that pre-processing and extracting matching ingredients from method context can be achieved.

During corpus construction, we filtered out all articles that don't fit into the three prominent structures, which includes ill-formed articles and gallery-style articles. This leaves us a corpus with 9,748 files out of 9,751 articles from wikiHow, and 7,559 out of 8,302 from eHow – a total size of 17,307.

4.3 Indexing of Recipe Documents

The recipe search part of our model still largely depends on a search index. We work with Apache.lucene to index and search.

Though, the index construction is rather simple. Each lucene document consists of 5 fields: (1)title, (2)id, (3)ingredients, (4)categories, and (5)file path. We did not store the main part of the articles, namely, the recipe instructions also into its own field, mainly because the index only serves the searching step, but not result interaction. Whether users want to search via ingredients or directly via a course name, the fields above already satisfy their need. However, because our raw data does not contain recipe tags, and categorizations have number restrictions (each recipe has 2-3 usable categories), search by dish category is also restricted. The statistics of categories can be seen in Table 4.4.

We only use the Lucene index for the Search step (Section 3.1) whereas the Result Interaction (Section 3.2) works directly off the JSON documents. Both are detailed further in Chapter 5.

4.4 Statistics

Our corpus consists of 17,307 JSON documents, with 9,748 from wikiHow and 7,559 from eHow. Table 4.2 summarizes the properties of corpus files. Having step-level abstraction is a prerequisite for a successful step-by-step cooking instruction of result interaction. Thus, all documents without it were considered ill-formed and were filtered out during corpus construction. As shown in the table, a drastic contrast can be observed regarding files that have sub-step-level information from both portals. This shows that eHow documents are much less likely to have a fine-grained structure.

Table 4.2: Overview of important file properties from both portals.

	eHow	wikiHow
Files with multiple parts	3569	4423
Files with step-level info	7559	9748
Files with sub-step-level info	755	9748
Files with ingredients	3963	8836
Parts per document	3	3
Steps per document	7	3
Sub-steps per document	1	1
Ingredients per document	4	2
Total file	7559	9748

As our data was collected from online how-to guides portals, most of the article titles are formulated as questions. An overview of 3-gram analysis of article titles is illustrated below in Table 4.3.

The most frequent prefix of the titles is "how to make", which covers an amount of almost the half of all articles. It is also obvious that "how to make" and "how to cook" are vastly more frequent than any other prefix. That's also part of the reason that we treat "make" and "cook" as stopwords during retrieval (see Section 5.1.1).

Table 4.3: Top 10 most common 3-grams of article titles.

3-gram	Impression
How to Make	8476
How to Cook	1409
How to Bake	343
How to Use	274
How to Prepare	210
How to Freeze	196
How to Store	149
How to Eat	133
How to Grill	132
How to Roast	106

To have a better understanding of our corpus composition, we inspected the article categories. Because the categorizations of wikiHow and eHow, again, are very different from each other, a separate overview of the two portals is shown in Table 4.4.

As can be seen in the table, the data we collected from wikiHow are all under the "recipes" category. While eHow also have e.g. "chicken recipes", "easy recipes", etc., articles categorized otherwise can also be recipes, so that distinguishing non-recipes from recipes is made difficult, as previously stated in Section 4.1. With regard to cuisines which can be a very important search parameter, wikiHow details them into different regions, while eHow only has a general category "world & regional food".

As already pointed out in Section 4.3, in stead of unifying the categories from both portals, we suggest automatic recipe classification or tagging [Kicherer et al., 2018].

Table 4.4: Article category overview. Because of the large number of third level categories on both portals (wikiHow:105, eHow:122), only the top 20 of each are shown. No direct relation between levels of categories is implied in this table.

1st-level	Count	2nd-level	Count	3rd-level	Count
		wikiHow			
Recipes	9748	Baking	2112	Cakes	761
		Fruits and Vegetables	1056	Cookies and Biscuits	480
		Desserts and Sweets	996	Poultry Recipes	401
		World Cuisines	983	Asian Cuisine	311
		Meat	937	Indian Cuisine	274
		Chocolate and Candy	531	Breads	273
		Jams Preserves and Condiments	441	Ice Cream and Frozen Desserts	263
		Vegetarian	399	Dessert Pies	238
		Sandwiches and Quick Meals	399	Beef and Lamb	222
		Fish and Seafood	397	Vegan	217
		Eggs and Dairy	335	Central and South American Cuisine	209
		Pasta and Noodles	318	Potato Dishes	186
		Salads	263	Fruit Desserts	177
		Rice and Beans	205	Sauces	172
		Soups	122	Pork	140
		Specialty Diet Recipes	112	Pizza	138
		Cereal Grains	32	Pastries	107
				Puddings	92
				European Cuisine	91
				Muffins	86
		eHow			
Food & Drinks	7559	Cooking & Baking	2529	Cooking Techniques	684
		Main Dishes	2026	Meat Recipes	682
		Appetizers, Soups & Salads	1097	Produce & Pantry	608
		World & Regional Cuisine	857	Cooking Basics	337
		Drinks & Cocktails	375	World & Regional Food	261
		Desserts	297	Appetizers	212
		Holiday Recipes	131	Fish Recipes	207
		Breads & Breakfasts	107	Baking Techniques	195
		Healthy Recipes	90	Cookware	187
		Wine	44	Grilling	175
		Diet & Nutrition	6	Snacks	171
				Chicken Recipes	168
				Baking Basics	167
				Easy Recipes	151
				Seafood Recipes	148
				Vegetable Recipes	137
				Shellfish Recipes	120
				Soup Recipes	110
				Other Drinks	108
				Cheeses	107

Chapter 5

Conversational Recipe Search Prototype

Voice agents provide users with the possibility of hands-off convenience, thus, agents can be a very useful asset when users are cooking, both hands normally being preoccupied by cutting or being sticky so that touching a device may cause extra trouble. We introduce our conversational cooking assistant prototype in this chapter that implements our interaction model. The prototype is developed with Alexa Skills Kit as an Alexa skill. It can not only help users find desired recipes, but also assist them with cooking recipes by providing step by step instructions, which corresponds with the two parts of our proposed model in Chapter 3. Our aim is to provide a personalized human-like assistance. Although the prototype is not yet a finished product, it serves already as a promising basis of further development.

The skill consists of a total of 37 Intents, 9 of which are Amazon built-in intents with or without our custom extension on sample utterances depending on the individual cases. An overview of all intents and their consistencies can be seen below in Table 5.1. Unlike the intent classes we defined in Table 3.1 that include intents from both sides, Alexa Skills Kit specifies only user intents. More about the Skills Kit and its terms were explained Section 2.2.

Also, many intents listed here can fall into multiple defined classes depending on the context or contain several classes in one turn. For example, of the user utterances that are recognized as SearchWithIngredIntent, if users say "I want something with rice and eggs" at the start of a session, it is OQ (Table 3.1); however if they say this during the search, it is FD from the users' side. Furthermore, "no, I want..." is a combination of NF and FD. This corresponds with the multiple moves of Trippas et al. [2018].

Table 5.1: Overview of all prototype intents and statistics of their utterances and slots.

Intent	Utterances	Slots	Description
HowQueryIntent	22	3	Users ask how to cook a specific course.
SearchWithIngredIntent	16	2	Users start a search with desired ingredients, or add ingredients as extra parameter to the current query.
SearchWithoutIngredIntent	19	2	Users tell Alexa the ingredients that should be excluded from the search.
SearchSpecifyIngredIntent	7	2	Users start a search or modify the query with both desired and undesired ingredients
SearchParamIntent	15	2	Users start a search or modify the query with extra category restriction or cooking method restriction.
AlwaysExcludedIngredIntent	7	1	Users tell Alexa what they don't eat. The ingredients will be stored and excluded in all searches regardless of sessions.
LiveInstructionIntent	3	-	Users signal Alexa to start live instruction for cooking the found recipe.
ReadIngredientsIntent	9	-	Users ask Alexa the ingredients without specifying which part.
MoreDetailIntent	4	-	Users request for more detail for a cooking step. This signals the system to read sub-steps.
LessDetailIntent	6	-	Users request for less detail for a cooking step. This signals the system to stop reading sub-steps.
ReadCertainIngredListIntent	17	4	Users ask Alexa to read the ingredients of a certain method or part of the found document.
WaitIntent	3	2	Users signal Alexa to pause and wait for a certain period of time.
ReadIntroductionIntent	7	-	Users ask for the found document's introduction.
InstructionForMethodIntent	17	3	Users ask for the live instruction of a certain method or part.
AskSpecificQuantityIntent	6	1	Users ask for the quantity of an ingredient in the recipe.
AskTimeIntent	12	2	Users forget how long they should cook something that was said before and thus asks Alexa.
AskConfirmIngredIntent	7	1	Users ask Alexa if a certain ingredient is present in the recipe.
PrepareIngredientsIntent	8	-	Users signal Alexa to help prepare the ingredients for cooking.
MealIdeaIntent	11	-	Users don't know what to cook and ask Alexa for recommendation.
SupermarketIntent	9	-	Users answer Alexa that they don't mind going shopping for the recipe.
StayHomeIntent	7	-	Users answer Alexa that they don't want to go shopping.
IHaveIntent	2	1	Users tell Alexa what ingredients they have at hand one by one.
SignalListCompleteIntent	3	-	Users signal that the list of on-hand ingredients is complete.
NewSearchIntent	2	-	Users want to search anew. Alexa will clean all parameters.
SummarizeMethodIntent	6	-	Users ask for short information of the documents's methods.
AdventurousIntent	1	-	User answers agent that they are being adventurous and willing to try new genres.
RequestCurrentQueryIntent	5	-	Users lose track of the current query and ask Alexa.
RemoveQueryParamIntent	2	1	Users adjust the query by removing parameters.
AMAZON.PreviousIntent	3	-	
AMAZON.NextIntent	2	-	
AMAZON.RepeatIntent	5	-	
AMAZON.YesIntent	1	-	
AMAZON.NoIntent	9	-	
AMAZON.StopIntent	2	-	
AMAZON.HelpIntent	-	-	
AMAZON.CancelIntent	1	-	
AMAZON.FallbackIntent	-	-	

5.1 Recipe Search

In this section, we first introduce the functionalities and interactions of the searching part of our prototype. Later, in the subsections, we will explain the search algorithms. The intent classes were defined in Table 3.1.

The search can help users in four different scenarios: (1) they know exactly what course they want to cook (complete item), for example, mushroom risotto without wine; (2) they have a preferred category (partial item), for example Indian; (3) they don't know what to cook but have preferred ingredients; and (4) they have no idea what to cook at all and want inspiration.

The first three are directly mapped to (1) title search, (2) category search and (3) ingredient search in the query formulation phase (OQ). The system will directly provide results one at a time according to the ranking score in descending order (PA). If the user gives NF, the next ranked document will be returned, so on and so forth, until the user gives PF, which starts the cooking instruction that we'll explain in the next section.

On the first NF, the agent will inform the user that they can add further parameters to the search (FD) to help the user use the system more efficiently. Users can add parameters of different fields to the query at any time. Also, to avoid giving too much information to cause stress for users, only the title of the recipe document is replied back to them. Users may ask questions regarding required ingredients (IR), which we consider as one of the most important piece of information for users, to help with their decision. Another piece of important information is the required cooking time which users are unable to ask about in this version, unfortunately, because of the lack of it in our source data. An example dialog is shown below in Figure 5.1.

In scenario (4), the agent will ask the users some questions after the OQ about their preferences to find out or help them find out their information need (IR). Here, the first question from the agent is if the user mind going extra shopping. Then, according to the user's answer, the agent will either help the user find recipes with available ingredients or suggest recipes according to the user's search history. The former then becomes an ingredient search, while the latter presents itself as a high personalization in our prototype.

The system logs users' called intents, title searches, included and excluded ingredients, category searches and confirmed recipes that the user really got to cook. It summarizes the most preferred items using a simple word count

U: **Open recipe search.**

A: Hi! How can I help you with cooking?

U: **I want to cook risotto.**

A: I found an article: The History of Risotto. Is this what you're looking for?

U: **No.**

A: I found another one: How to Make Risotto. Is this what you're looking for? By the way, you can also tell me the ingredients that you want or don't want to cook with, if there's any.

U: **I want to cook with mushroom.**

A: Here! I found this: How to Make Mushroom Risotto. Is that what you're looking for?

U: **Does it have wine in it?**

A: Yes. wine is in the recipe. So... You still want it?

U: **No.**

A: I found another one: How to Make Risotto. Is this what you're looking for?

U: **Is mushroom in it?**

A: Yes. mushroom is in the recipe. So... You still want it?

U: **Yes.**

Figure 5.1: A real aimed-searching dialogue using the prototype. It illustrates scenario (1) where the user knows what they want to cook. U: user, A: agent.

per division and make it known to the user. Next, the user may either stick to their usual preferences or be adventurous and choose from randomly suggested genres that they've never cooked before, pulled from a list of categories extracted from the source data. Once the user chooses a recommended genre, a category search will be carried out. An example of it is illustrated in Figure 5.2.

A: Hi! How can I help you with cooking?

U: **I don't know what to cook today..**

A: Do you mind going to the supermarket or would you rather use the ingredients on hand?

U: **I don't mind going out.**

A: You often search for recipes that include rice, and risotto recipes, and Indian recipes, Which would you like: one of these or are you feeling adventurous today?

U: **Adventurous.**

A: Okay, let's see. How do you think of chicken recipes or vegetarian recipes? Say, I want bla, so I can understand you better.

U: **None of these.**

A: How do you think of seafood recipes or vegetable recipes?

U: **Seafood recipes.**

A: I found an article: How to Make Seafood Stew. Is this what you're looking for?

Figure 5.2: A real no-aim search dialogue using the prototype. It shows scenario (4) where the user requests recipe recommendation. U: user, A: agent.

5.1.1 Retrieval

The intents involved in the retrieval process are: HowQueryIntent, SearchWithIngredIntent, SearchWithoutIngredIntent, SearchSpecifyIngredIntent, SearchParamIntent, AlwaysExcludedIngredIntent and RemoveQueryIntent.

HowQueryIntent has three slots: action, article and object. Action stands for the verb, the cooking method such as "stew" or "fry" that we consider important as users' need. Article is the collection of words like "a", "the", "some" and so on that convey no important information. Object is the food or course name. We take the recognized action and object words as a query in title field of the indexes. Although, "cook" and "make" are two verbs that we exclude from the query as stopwords, because they are too general and are proven to negatively affect search result quality.

For the ingredients-relevant query intents, the desired and undesired ingredients are read as separated slots and parsed into a lucene query in the ingredient field, with each undesired ingredient coupled with a NOT-operator that ensures their definitive non-appearance in result documents. However, a search with NOT-operator is restricted due to lucene's functionality, namely, if no other operators exist in the same standalone query, no result will be found and the search is unsuccessful. This means, if users want to exclude ingredients, they have to tell Alexa at least one ingredient to include even if they've given a title query. Similarly, the category slot words of SearchParamIntent are parsed into a query in the category field.

If search parameters in different fields are given, the standalone lucene queries of those fields are built into a BooleanQuery, a collective query object that combines multiple queries of different types. All words in a standalone query are connected with an OR-operator.

Cooking Ontology

We expect that many queries may contain food or ingredient words that are more general, such as "I would like to have meat in the recipe", where chicken, beef, pork, etc. should all be included. This need could be easily met if we had more categories or tags for each recipe in our source data, so that all recipes tagged with meat can be filtered out. The extensively researched cooking ontology and recipe classification could be utilized in our case to pre-process the data [1] [Nanba et al., 2014][Kicherer et al., 2018][Chung, 2012].

---

[1] `http://www.foodsubs.com/`

However, the main focus of this thesis being conversational search, we didn't delve into recipe classification, but rather took an experimental approach directly using WordNet [2], a large English lexical database where words are grouped into sets by lexical and semantic relations.
With the hierarchy structure provided by WordNet, we can simply derive hyponyms of words using the Java WordNet Library [3]. Hyponyms are the sets of words that are more specific, e.g. pork is a hyponym of meat.
Aside from extracting hyponyms, the question of which words shall we extract hyponyms from remains, but this is left for future work. For our experimental purpose, we created a lexicon of words that are hyponyms of "meat" considering real-time extraction. If "meat" should be included in ingredients, the lexicon will be integrated into the BooleanQuery as a BoostQuery with the boost value 1 divided by the number of words in lexicon to avoid weight influence on the results; if it should be excluded, each word is given a NOT-operator and directly appended to the ingredient query.

5.1.2 Ranking

Currently, our prototype uses the default ranking function, the Okapi BM25, a sophisticated version of TF-IDF. In a combined BooleanQuery, each field has the same weight (boost value = 1.0), except the aforementioned hyponym case.

5.1.3 Single-Turn Q&A

As spoken of above and depicted in Figure 5.1, users may ask questions regarding required ingredients to help them identify desired recipes. This is done without the indexes, but with a simple algorithms that finds matches of the spoken ingredients words or phrases with all the ingredient lists of the document (some documents may include more than one list of ingredients if they include multiple methods or parts).

5.2 Cooking Instruction

The whole instruction part corresponds with the multi-turn navigatable interaction of our model that was introduced in Section 3.2.2.

---

[2] `https://wordnet.princeton.edu/`

[3] `https://web.stanford.edu/class/cs276a/projects/docs/jwnl/javadoc/`

There are two possible responses after a user gives positive feedback to a result: (1) if the recipe has more than one part or method, the agent will read a summary of methods - in this case, the method names; and (2) if the recipe is without parts/methods, Alexa will read the ingredients instead, if ingredients exist, but without the quantities or any additional information. Both give the user an overview of the recipe.

The user will then be informed that they can ask for live instruction, then the information unit exploration phase will start. If a method has a corresponding ingredients list, the ingredients will be read at first, also without quantities. Users may ask for the quantities or indicate Alexa that they want to prepare the ingredients. The system will read the ingredients one by one and always wait for users' cue to continue to the next one.

Analogous to ingredients preparation, each step of the recipe will be read on users' directives. By default, the agent reads the step-level only, with the addition of tips and warnings if they exist. On users' MoreDetailIntent, the system will toggle the detail level and read both the step and the step's substeps. This works also vice versa. Furthermore, users are allowed to jump to a certain method, since many documents contain more than one method to cook something, which can be seen as recipes inside recipes.

An example of cooking instructions is shown on the next page in Figure 5.3.

5.2.1 Similarity Computation for Information Division Retrieval

As stated in Section 3.2.2, the user may choose a browsing destination, an information division from the document, for example ingredients, or a certain method. Yet, after users have gained an overview of available methods, we expect them to request a method with different approaches: (1) naming the index, (2) referring by name, and (3) saying both the index and name.

Case (1) is quite straightforward. We retrieve the method directly by the given index. Though, in case (2) and (3) when users refer to a method by its name, it gets more complicated, because they wouldn't necessarily repeat the exact name. E.g. in the example shown in Figure 5.3, of all the methods to make a chocolate cake, they might refer to Method 6 Vegan Chocolate Cake as "the vegan one". A similar problem exists for matching ingredient lists to methods as well.

U: **How to make a chocolate cake?**
A: I found an article: How to Make a Chocolate Cake. Is this what you're looking for?
U: **Yes.**
A: There are more than one method to do it. Method 1 Basic Chocolate Cake, Method 2 Moist and Fluffy Chocolate Cake, Method 3 Eggless Chocolate Cake, Method 4 Milkless Chocolate Cake, Method 5 Gluten-Free Chocolate Cake, and Method 6 Vegan Chocolate Cake. You can ask me for live instruction if you want to cook this recipe.
U: **Jump to method two.**
A: Method 2 Moist and Fluffy Chocolate Cake. We're gonna need: all purpose flour, salt, baking powder, baking soda, unsweetened cocoa powder, sugar, hot coffee, canola oil, buttermilk, eggs, and vanilla extract. So, here's how it goes: first, Prepare to make the cake. Preheat the oven to 325 Fahrenheit or 162 Celsius. Grease and flour two 9-inch round baking pans.
U: **How much of each ingredient do i need?**
A: First, 2 cups all-purpose flour, sifted . If you're finished with preparing this, say next.
U: **Next.**
A: 1 teaspoon salt
U: **Next.**
...
A: 1 teaspoon vanilla extract. This is the last ingredient.
U: **Next.**
A: Prepare to make the cake.
U: **Tell me more, please.**
A: Preheat the oven to 325 Fahrenheit or 162 Celsius. Grease and flour two 9-inch round baking pans.
U: **Next.**
...

Figure 5.3: An excerpt of a real cooking instruction from the prototype from a document that contains multiple recipe methods. U: user, A: agent.

There are different approaches to similarity computation. We examined edit distance metrics and relevance metrics. The former computes the dissimilarity between two strings according to the minimum number of operations to transform one string to another, while the latter is usually used by search engines and measures the relevance between tokenized queries with documents according to terms.

Our goal is to, for example, match *grilling chicken breast* with *grilled chicken breast* rather than *baked chicken breast*, where *grilling* should have more similarity with *grilled* than *baked*; and match *microwaved* with *microwaved scrambled eggs* rather than *scrambled.* We resolve the problem with a combination of Levenstein's algorithm of edit distance metrics and longest common substring in the end for their inexpensive computation yet nonetheless efficiency. Our algorithm prioritizes the exact same string with a difference of 0, then longest substring, and lastly differences larger than 0.

Case (3) when users say both the index and name, because people usually remember meaningful content better than plain numbers in their working memory, we allow falsely remembered indexes by prioritizing the names. If the system fails to find a match with the given name, the uttered index will be used for retrieval.

5.2.2 Machine Comprehension for Q&A

In Section 3.2.2, it was stated that users are allowed to give information requests at any time. As mentioned previously as well in Section 3.1.3, questions about the document often requires machine reading comprehension.

In real life, users might ask a wide range of questions regarding the recipe. However, although Alexa provides us with a powerful development framework that eases our research to a great extent, it is also restricting. All the questions directed at Alexa have to be categorized into intents. To avoid misinterpretation of different intents, a certain amount of distinctions between intents have to be established. So, if we define an intent with many wildcards (slots) to mimic a number of different types of questions, many other user intents will be falsely classified as this one. Moreover, the developers have no access to the intent classification process. These directly result in a limited amount of different questions that can be understood inside the Alexa skills' framework.

In this manner, the questions that are covered in our prototype are those regarding previously mentioned (1) cooking time, and (2) ingredients' quantity. We achieve this by adopting the AllenNLP Machine Comprehension tool's API [4]. A question is reconstructed from the slot phrases we collect from Alexa (developers have no access to the whole user utterance, only the filled slots). We assume that users only instruct the agent to continue when they finished cooking the previous step, so the machine comprehension is based on what was previously read to maintain the context. The question and text snippet are sent to AllenNLP and then the answer is collected and directly returned to users. An example dialogue is illustrated in Figure 5.4.

---

[4] `https://allennlp.org/models`

A: Add the chicken to the marinade. Cover the bowl and set it in the refrigerator. Allow the chicken to marinate for at least one hour. For best results, do not allow the chicken to marinate more than one day. This is the last step of this part.
U: **How long should I marinate the chicken again?**
A: At least one hour.

Figure 5.4: An example dialogue excerpt of Q&A using machine comprehension. U:user, A:agent.

5.3 Conversational Search Properties of the Prototype

Here, we examine the conversational search properties defined by Radlinski and Craswell of our prototype. The terms were introduced in Section 2.1.

User Revealment

To help users express or discover their true information need, our prototype system prompts for feedback and gives guidance on how users can formulate their queries. For instance, for users' negative feedback on a result, the system tells them that it's possible to restrict ingredients. Preferences on food or ingredients to be always excluded are also stored long-term.
Furthermore, by analysing user logs, our system is able to make personalized recommendation.

System Revealment

The system will inform users about its capabilities shortly if users ask for help. By requesting a rating or critique and partial item choices from users, the system both demonstrates the ways in which it can partition and refine the search space.
Alas, if users ask for something that's not computed, the Alexa cloud will either recognize that the utterance can't be mapped into any existing intent and gives correspondingly a feedback, or categorize the request falsely into an existing intent as mentioned before. This is due to our development platform's restriction.

Mixed Initiative

A dialogue can only be initiated by users invoking the skill. However, inside a dialogue, both the user and the agent may take the initiative to give prompts depending on the completeness of the revealed user information need. An example of the agent taking the initiative is asking questions in order to recommend recipes better.

Memory

Radlinski and Craswell suggested two distinct roles of memory in conversational search settings: (1) the system recalling past statements by default, and (2) users' ability to refer explicitly to past statements. The first is addressed both implicitly and explicitly in our prototype. The continuation of a conversation implicitly requires the continuation of a user's information need and the previous state. Explicitly, our system has the memory of previous searches through logs.
The second is addressed through our implementation of machine comprehension where users are able to ask questions about the previous step.

Set Retrieval

Our system does not have the function to find complementary items. This could however be implemented for related recipe recommendation in the future.

5.4 Interaction Approaches of the Prototype

The interaction approaches of the theoretical framework were introduced in Section 2.1, along with an overview in Figure 2.1. We now analyze our prototype with respect to said interaction approaches.

Null System - Free Text User

Alexa itself is such a system that allows many possible requests. However, because our skill is cooking-oriented, not any query can be entered, and is itself not a null system.

Partial Item System - Pref/Rating User

A user may be presented with partial information about matching results in various ways. This approach is used for the system to either confirm something that has been inferred, for example in our system "I found ... Is this what you're looking for?", or ask preference, such as "Which one would you like, ... or ...?"

Partial Item System - Critique User

This approach presents a more powerful retrieval paradigm where the user may provide a richer answer than a simple score or preference. In the simplest case, fielded search provides users with a selection of known fields and users may select or specify ranges for any property they desire, which is common in online shopping scenarios [Radlinski and Craswell, 2017], but this feature is not supported in our prototype. In other settings a user is presented with specific individual facet values and the agent allows users to clarify, rather than allowing only a simple yes or no. For instance in our system, to answer the prompt "I found this: How to make chili flavored chips. Is this what you're looking for?", the user may reply "No, I don't eat chili", which will then taken as an additional parameter to the search.

Partial Item System - Free Text User

This approach corresponds with slot filling where the system asks the users to fill in a particular aspect of an information need. For example, Alexa may ask the user to respond in a structured way to fill in the method slot by "I can understand you better if you say: ingredients for method bla."

Complete Item System - Pref/Rating User

Classic approaches to recommendation often request ratings of items to learn a user model for further recommendations. These may be absolute rating requests or preference requests. This is not supported in our system yet, but we can imagine asking users for their rating after finishing the instruction, both on the helpfulness of the instructions and the food itself. A learning algorithm has to be designed to utilized the information gathered to improve the result ranking so, that it's also customized.

Complete Item System - Critique User

In this case, a system may select a given item, then allow the user to refine their information need anchoring of the properties of the item. The query refinement in our prototype resembles this approach where a complete item is presented as a result and further metadata or details can be requested and new query parameters can be recognized during users' incremental critiques [Reilly et al., 2005].

Chapter 6

Conclusion and Future work

In this work, we put forward a first human-like conversational information retrieval model that specifies two information retrieval purposes: document search, and result interaction. As voice agents are often used in hands-off situations, we implemented our proposed model to build a conversational recipe search prototype as an Amazon Alexa skill based on a data collection of 17,307 recipe documents. We further analyzed the prototype with the theoretical framework of Radlinski and Craswell. In this chapter, we summarize our research and its findings in Section 6.1, and present an evaluation study proposal for our recipe search prototype and possible areas of improvement and further research in Section 6.2 Future Work.

6.1 Contributions

We started our research by asking the following research questions: (1) How should a new information seeking model for speech-based conversational systems look like? (2) What are effective techniques to present results using audio? and (3) How to evaluate the effectiveness of a conversational system?

To answer these questions, we investigated related work in Chapter 2 and proceeded to construct a conversational information retrieval model in Chapter 3. The model was presented in two parts, conversational search and conversational result interaction. We discussed conversational result presentation and focused on avoiding user cognitive overload as well as enabling users to feel in control. We defined a collection of intent classes from both user-side and agent-side to illustrate examples of intent flows of the model.

Based on the corpus that we presented in Chapter 4, we implemented a recipe search prototype that is also able to give cooking instructions in Chapter 5 using the proposed model to demonstrate its applicability and capability. The prototype itself involves several extend-able areas worth researching regarding conversational recipe search, which we will discuss more in the next section. In order to answer the final research question, we evaluated the prototype with Radlinski and Craswell's theoretical framework.

The evaluation of both the prototype and the model requires exhaustive real-life implementation and user studies. Our work does not give a final answer to the research questions, but we believe it makes a plausible step towards the answers.

6.2 Future Work

In this section, we first propose an evaluation study for our conversational recipe search prototype, as it is essential for providing an insight into how well the system accomplishes its goal and how it can be improved. Then, in section 6.2.2, we discuss further future work.

6.2.1 Evaluation Study Proposal

There are two evaluation approaches to a spoken language system according to Polifroni et al. [1998], one being component evaluation that examines the behavior of each part of the system; the other one judges system behavior by inspecting each query/response pair, in our case, intent/response.

The main components of the whole prototype system, regardless if the component is accessible from the developers' side, are: speech recognition, parsing, intent classification (understanding), and system response. While the first three components are carried out in the Amazon cloud service and thusly inaccessible, the intent classification can still be improved to an extent by altering the defined sample utterances or intents.

We suggest a combination of both evaluation approaches mentioned above. Intent classification and system response components are examined separately, where the latter shall be further investigated using the second approach. In the following sub-sections, each will be explained in detail.

Intent Classification

To give appropriate responses to user requests, the understanding of user intents is essential. The evaluation of intent classification should clarify these questions: (1) Are the user intents classified correctly? and (2) To what extent do the defined intents encompass possible user requests?

We suggest having a collection of participants that have different experience levels with voice agents from a scale of never contacted to multiple interactions per day in order to maintain the diversity of utterances. Each should be given the same set of task scenarios for control purposes.

Since the full texts of user utterances are not accessible, the voice interactions between participants and the agent should be recorded. Meanwhile, each classified intent should be logged. From the records, not only the exact user utterances are obtained, but also their possible negative feedbacks if the system gave undesired responses. By comparing the records with corresponding logs, false classifications can be identified and utterances that trigger the FallBackIntent can be noted (utterances that cannot be categorized into any existing intent).

A manual classification of the falsely classified utterances should be carried out to enable the detection of similarly defined intents. These intents can then be carefully compared in order to discover the reason for the confusion, be it too similar sample utterances or slots distribution. An improvement can then be performed by distinguishing these intents further from each other.

Last but not least, an observation of un-categorizable utterances may reveal user information needs that are not yet included.

System Response

By evaluating the system response, we can explore answers to (1) How well does the system accomplish its goal in helping users? and (2) How easy is the system to learn?

Inspecting each intent/response pair is meaningful, but would not suffice to judge the system behavior of our prototype, since it includes more than simple single-turn Q&A. The context is therefore important and must be considered. Hence, in addition, we would like to examine the interaction in small modules, for example recipe recommendation module, recipe search module, ingredients preparation module, etc.

To understand user satisfaction and to answer the two questions stated before, the participants will be asked to rate (1) how useful was the system for the tasks, (2) how difficult it was to interact with the agent, and (3) their feeling on a scale from extremely frustrated to completely satisfied. They will be also asked to give reviews on the system about what was impressively satisfying, what hindered the participants' tasks and what could be improved.

Not only can we collect these qualitative user data, but also quantitative, which includes e.g. time on task, task completion or success rate. Expert review can be employed as well. However, all the measurements mentioned above are not only relevant with the interaction effectiveness, but also retrieval effectiveness. Thus, we suggest to first evaluate and improve the retrieval process based on the classic Recall and Precision before any interaction evaluation in order to avoid misinterpretation of measured data.

6.2.2 Other Future Work

Our proposed model can serve as an applicable foundation for further conversational information retrieval systems. A possible future task would be developing methodologies for a direct evaluation of such an abstract model in stead of evaluating a derived prototype system.

Through our implementation of the recipe search prototype, we came upon several research questions with respect to recipe search. For example, is it possible to extract cooking duration information from the verbs or phrases presented in a recipe. An experienced cook may know that french fries need to cook for about 5 minutes, whereas chicken thighs around 15 minutes – can this be automatically distinguished through learning?

Another research direction may be the evaluation of present machine comprehension models. Since most are trained with scientific articles, how do they perform on recipes exactly? Other than that, step segmentation of a recipe can also be worth researching.

Without the restriction of our development platform, the prototype itself could be extended or improved in the following aspects: integration of cooking ontology, customization of system properties such as how many results should be presented at once, user question handling during a conversation, preference learning through intents, etc.

To conclude, the conversational Information Retrieval model that we presented can be applied without domain-specific restrictions. Together with the recipe search prototype, it provides a starting point of many possible directions of research that holds large potential.

Bibliography

James Allan, Bruce Croft, Alistair Moffat, and Mark Sanderson. Frontiers, challenges, and opportunities for information retrieval: Report from swirl 2012 the second strategic workshop on information retrieval in lorne. *SIGIR Forum*, 46(1):2–32, May 2012. 1

Nicholas J Belkin, Colleen Cool, Adelheit Stein, and Ulrich Thiel. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert systems with applications*, 9(3):379–395, 1995. 2.3

Young-joo Chung. Finding food entity relationships using user-generated data in recipe service. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2611–2614. ACM, 2012. 5.1.1

Carsten Eickhoff, Jaime Teevan, Ryen White, and Susan Dumais. Lessons from the journey: a query log analysis of within-session learning. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 223–232. ACM, 2014. 1

Hideo Joho, Lawrence Cavedon, Jaime Arguello, Milad Shokouhi, and Filip Radlinski. First international workshop on conversational approaches to information retrieval (cair'17). In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1423–1424. ACM, 2017. 1

Tom Kenter and Maarten de Rijke. Attentive memory networks: Efficient machine reading for conversational search. *CoRR*, abs/1712.07229, 2017. 1

Hanna Kicherer, Marcel Dittrich, Lukas Grebe, Christian Scheible, and Roman Klinger. What you use, not what you do: Automatic classification and similarity detection of recipes. *Data & Knowledge Engineering*, 2018. 4.4, 5.1.1

David Kirsh. A few thoughts on cognitive overload. *Intellectica*, 1(30):19–51, 2000. 3.2.1

J Lai and N Yankelovich. Speech interface design. 12 2006. 3.1.3

Aleksi Melto, Markku Turunen, Anssi Kainulainen, Jaakko Hakulinen, Tomi Heimonen, and Ville Antila. Evaluation of predictive text and speech inputs in a multimodal mobile route guidance application. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pages 355–358. ACM, 2008. 3.1.3

George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956. 3.2.1

Hidetsugu Nanba, Yoko Doi, Miho Tsujita, Toshiyuki Takezawa, and Kazutoshi Sumiya. Construction of a cooking ontology from cooking recipes and patents. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 507–516. ACM, 2014. 5.1.1

Joseph Polifroni, Stephanie Seneff, James Glass, and Timothy J Hazen. Evaluation methodology for a telephone-based conversational system. In *Proc. LREC*, volume 98, pages 43–50, 1998. 6.2.1

Chen Qu, Liu Yang, W. Bruce Croft, Johanne R. Trippas, Yongfeng Zhang, and Minghui Qiu. Analyzing and characterizing user intent in information-seeking conversations. *CoRR*, abs/1804.08759, 2018. 3.1

Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17, pages 117–126. ACM, 2017. 1, 2, 2.1, 2.1, 3.1.1, 5.3, 5.3, 5.4, 6, 6.1

James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Incremental critiquing. *Know.-Based Syst.*, 18(4-5):143–151, August 2005. 5.4

Nuzhah Gooda Sahib, Anastasios Tombros, and Tony Stockman. A comparative analysis of the information-seeking behavior of visually impaired and sighted searchers. *J. Am. Soc. Inf. Sci. Technol.*, 63(2):377–391, February 2012a. 2.3

Nuzhah Gooda Sahib, Anastasios Tombros, and Tony Stockman. A comparative analysis of the information-seeking behavior of visually impaired and sighted searchers. *Journal of the Association for Information Science and Technology*, 63(2):377–391, 2012b. 1

Adelheit Stein and Ulrich Thiel. *A conversational model of multimodal interaction.* GMD, 1993. 2.3, 2.2

Johanne R. Trippas. Spoken conversational search: Speech-only interactive information retrieval. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, CHIIR '16, pages 373–375. ACM, 2016. 1

Johanne R. Trippas, Damiano Spina, Mark Sanderson, and Lawrence Cavedon. Results presentation methods for a spoken conversational search system. In *Proceedings of the First International Workshop on Novel Web Search Interfaces and Systems*, NWSearch '15, pages 13–15. ACM, 2015. 3.1.3

Johanne R. Trippas, Damiano Spina, Lawrence Cavedon, Hideo Joho, and Mark Sanderson. Informing the design of spoken conversational search: Perspective paper. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*, CHIIR '18, pages 32–41. ACM, 2018. 2.3, 2.3, 3, 5

Alexandra Vtyurina, Denis Savenkov, Eugene Agichtein, and Charles L. A. Clarke. Exploring conversational search with humans, assistants, and wizards. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 2187–2193. ACM, 2017. 2.3

Marilyn Walker and Steve Whittaker. Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, ACL '90, pages 70–78. Association for Computational Linguistics, 1990. 2.3

Nicole Yankelovich, Gina-Anne Levow, and Matt Marx. Designing speechacts: Issues in speech user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 369–376. ACM Press/Addison-Wesley Publishing Co., 1995. 3.2.2