Knowledge Graphs and Interactions in Conversational Agents

Politecnico
di Bari

DEPARTMENT OF ELECTRIC AND INFORMATION ENGINEERING
ELECTRICAL AND INFORMATION ENGINEERING PH.D. PROGRAM
SSD: ING-INF/05 INFORMATION PROCESSING SYSTEMS

Final Dissertation

Knowledge Graphs and Interactions in Conversational Agents

by

Giovanni Maria Biancofiore

Supervisor
Prof. Tommaso Di Noia

Coordinator of the Ph.D. Program

Prof. Mario Carpentieri

Course n°35, 01/11/2019 - 31/10/2022

Dedicated to all my families,
To my uncle Franco,
A fighter like I've never seen,
And to my little dog Luna,
Who will always shine bright in the sky.

Abstract

With the spread of online services and platforms, there is a growing need for methods that grant the fruition of today's digital world in accessible and natural ways. In a society steeped in the age of big data and computing reality, highly specialized knowledge is often required of people to enjoy these virtual benefits. Unfortunately, such wisdom is very challenging to acquire, also for technicians experienced in the Informative Systems domain. This field branches out into consistently different areas regarding goals and the skills required to achieve them. Moreover, individuals would have to handle vast and complex information shown in a machine-readable shape, which flows into unnatural and often incomprehensible approaches to interacting with digital systems.

Virtual Assistants (i.e. Google Assistant, Alexa and Siri) have taken root in solving these problems. In detail, Conversational Agents identify a fitting solution in allowing natural accessibility to online services for a broad audience. Such agents enable the interpretation and response to the users' statements in ordinary natural language, revolutionizing Human-Computer Interaction through Artificial Intelligence. Although Conversational Artificial Intelligence is not a recent research field, the goals it promotes are far from being fully achieved. Natural Language Processing and Machine Learning techniques have brought significant improvements in

the interaction quality offered by conversational agents, both in the ways and in the contents of dialogues. However, people still perceive a certain communication artificiality and difficulty of use when dealing with such systems. The main problems lie in the low adaptability of interaction and understanding of intentions and language towards users.

This thesis proposes to dissect the two most complex issues in implementing conversational agents, which regard their interactivity and capability to understand the audience's intents and languages. In particular, this dissertation collects all the research efforts made so far about the growth of dialogue systems, focusing on Question Answering Systems and Conversational Recommender Systems. It highlights how fundamental the modality and the ways of interacting with conversational agents are to improve user satisfaction. Moreover, we propose Knowledge Graphs based models to enhance the effectiveness of dialogue systems in understanding the human language and the users' personality. We have also explored some intuitions about the aspect-based sentiment analysis of text to retrieve people's opinions and emotional states on several topics. Our findings prove that deriving sentiment representations from the text will feed systems with richer informative content, which results in more versatile conversational agents regarding users' scenarios. Extensive experiments support our proposals, spreading through the previously mentioned analysis dimensions, which evaluate in summary the interactivity, the semantics and the sentiment featuring a conversational system. Finally, this dissertation presents a comprehensive literature review on Interactive Question Answering systems, which also take a picture of the several scenario and findings which still characterize the constant evolution of Conversational Agents.

Publications

The contents of this work are the result of the research efforts that have had space in some publications. Here a complete list of my publications is made available in which I contributed as a corresponding author.

[1] Vito Bellini, Giovanni Maria Biancofiore, Tommaso Di Noia, Eugenio Di Sciascio, Fedelucio Narducci and Claudio Pomo. "GUapp: A Conversational Agent for Job Recommendation for the Italian Public Administration." In: *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems.* EAIS, 2020, pp. 1-7.

[2] Giovanni Maria Biancofiore, Tommaso Di Noia, Eugenio Di Sciascio, Fedelucio Narducci and Paolo Pastore. "GUapp: Enhancing Job Recommendations with Knowledge Graphs." In: *Proceedings of the 11th Italian Information Retrieval Workshop 2021, Bari, Italy, September 13-15.* IIR, 2021.

[3] Giovanni Maria Biancofiore, Tommaso Di Noia, Eugenio Di Sciascio, Fedelucio Narducci and Paolo Pastore. "GUApp: a Knowledge-aware Conversational Agent for Job Recommendation." In: *Workshop Proceedings of the 3rd Edition of Knowledge-aware and Conversational Recommender Systems and the 5th Edition of Recommendation in Complex En-*

vironments co-located with 15th ACM Conference on Recommender Systems, Virtual Event, Amsterdam, The Netherlands, September 25. KaRS & ComplexRec, 2021.

[4] Giovanni Maria Biancofiore, Tommaso Di Noia, Eugenio Di Sciascio, Fedelucio Narducci and Paolo Pastore. "Aspect based sentiment analysis in music: a case study with spotify." In: *SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, April 25 - 29, 2022.* ACM, 2022, pp. 696-703.

[5] Giovanni Maria Biancofiore, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio and Fedelucio Narducci. "Interactive Question Answering Systems: Literature Review." *arXiv preprint arXiv:2209.01621* 2022.

[6] Vito Walter Anelli, Giovanni Maria Biancofiore, Alessandro De Bellis, Tommaso Di Noia and Eugenio Di Sciascio. "Interpretability of BERT Latent Space through Knowledge Graphs." In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 2022. p. 3806-3810.

Contents

List of Tables

List of Figures

1

Introduction

In the era of big data and digital platforms, society disposes of many online services designed and distributed to improve people's welfare and enforce physical infrastructures. Digital ecosystems like streaming platforms (e.g. Netflix, Spotify), e-commerce websites (e.g. Amazon, eBay) and social networks (e.g. Facebook, Twitter) fill today's virtual panorama. Such technologies further spread into people's everyday life with services employed to handle the most disparate needs, where healthcare, education and public/private administration identify only a few examples. Indeed, users interact with several digital services and devices to perform simple to complex tasks. Although online tools and intelligent machines have undoubtedly brought considerable benefits [158], the variety of these technologies may mislead people when engaging them, especially for executing sharp operations (e.g. online payments, searching for sensitive information, etc.). Thus, users usually prefer to rely on a simplified and more human-like system interaction [129].

Virtual Assistants (VAs) were born to solve this issue, filling the gap between specialized technicians and the average user. In detail, a VA is a software program able to assist people in performing some predefined tasks. It assumes different shapes depending on the type of task the VA implements (e.g. a smartphone application for daily functions like Google Assistant and Apple Siri [28], software for robotics in robot control [175],

etc.). However, all the VA forms share a common feature given by their Natural Language Interface (NLI) since it represents a concise and effective way for humans to interact with software and cloud services [49]. Indeed, its primary goal is to understand user intents expressed through Natural Language (NL) and reply to them appropriately, emulating a human conversation. This kind of interface in the literature takes the name of Conversational Agent (CA), whose summarized tasks are processing people's text or speech, understanding their needs, answering with automatically generated responses functional to the user purposes and maintaining a dialogue continuity.

In a few words, a CA [1] is an Artificial Intelligence (AI) program that originated to imitate human conversations using spoken or written natural languages over the Internet. Given their sophisticated role, such agents need many methodologies and tools characteristic of several Informative Systems fields to implement their scope. Therefore, CAs stand at the point of union of Human-Computer Interaction (HCI), Natural Language Processing (NLP) and AI subjects. In different application domains, CAs interactively engage users to accomplish their demands. For instance, they grant simplified access to healthcare structures and build people's anamnesis [50] or lead the seeking of information through NL dialogues [190]. In this way, CAs facilitate using digital services for a broad audience.

Nevertheless, we still know little about how users perceive CAs [245], although recent advances in Machine Learning (ML) and NLP techniques allow them to implement increasingly complex tasks [62]. How to design appropriate CAs for specific fields of application is still challenging [53], also because the majority of the publicly available CAs are "black boxes" where their inner workings contain highly-protected IP that is not accessible to the public [39]. Thus, there is no shared approach to good practices to design them. Current CAs are lagging in meeting users' expectations in terms of not being able to hold the conversations for a longer time [71], as well as answering correctly to the people's needs since they often implement less context-aware systems or keyword/pattern-based methods [1]. Moreover, they have broadly proven unfit for handling the users' emotions

and moods [168]. Summarizing the main issues that affect today's CAs, we can identify two main development topics: interaction and semantic understanding.

This dissertation aims to explore in detail the two previously mentioned topics, assessing which are the recent state-of-the-art developments and proposing novel and functional solutions for CAs. In particular, we focus on improving the CAs' interactivity by exploiting instruments like Knowledge Graphs (KGs) and the Aspect-Based Sentiment Analysis (ABSA). Furthermore, we outline how these findings benefit many downstream and upstream tasks strictly related to CAs, i.e. recommendation and Natural Language Understanding (NLU). Proving the existence of inner KGs regarding Language Models (LMs) would allow CAs to apply effective and well-established KG-based techniques on vectorial representations of text, thus broadening their interactive and understanding capabilities. Moreover, modelling entities and relations directly on word embeddings would provide a further way of retrieving and recognising aspects for the ABSA task.

Schematically, this thesis outlines our efforts to achieve the following contributions:

- The literature review of interactive systems dear to CAs, which highlights the main differences against non-interactive counterparts and provides a comprehensive taxonomy of their definitions, implementations and evaluation protocols;
- The empirical evidence of our proposed ABSA approach in retrieving a new kind of information from texts and the comparative investigation between conversational and classic systems in the recommendation scenario, which also state the advantages of interactivity;
- The presentation of KGs as a powerful tool for both the generation of dialogues for a recommendation task and the intuitive understanding of natural language text through LMs.

In the following sections, we provide an in-depth guide on the orga-

nization of this work, thus showing in detail the research questions that guided us toward the contributions of this thesis.

1.1 Thesis Organization

This dissertation proposes an exhaustive analysis of CAs, describing the panorama which characterizes the state-of-the-art. Our research focused on the interactivity of such dialogue systems, besides enlightening the benefits that KGs bring to the conversation and understanding skills of a CA. To this aim, we performed a literature review about interactive systems known as Question Answering (QA), which allows the delivery of fundamental and generic concepts regarding CAs and the recently adopted implementations to realize a high-performer interactive tool. Then, we observed the information gain given by sentiment-based text representation proposing a new approach based on ABSA, which would enrich the interactivity of any systems relying on Natural Language Processing (NLP). Thus, we confirmed the usefulness of conversational techniques through the comparative study of a domain-oriented Recommender System (RS), giving birth to the Conversational Recommender System (CRS) GUapp [16]. We finally moved to study the potential offered by KGs, equipping our previously proposed CRS with a knowledge-aware dialogue generator module, besides assessing the intrinsic existence of a KG semantic into the BERT [52] LM. Our findings lead to an improved generalization and understanding ability of systems that handle human languages, i.e., CAs.

The chapters composing this thesis are self-contained regarding definitions, the literature in which our research lay, proposed architectures, and contributions. In particular, Chapter 2 encloses all the theoretical background needed to deal with this dissertation, while Chapter 3 surveys the literature about interactive QA systems. Chapter 4 shows our research efforts on the sentiment-based representation of text, then Chapters 5 and 6 outline our proposal on CRSs. Finally, Chapter 7 holds our assessments on using KGs in the NLU task through LMs. Thus, Chapter 8 remarks on the conclusions reached by this work. Figure 1 summarizes the organization of

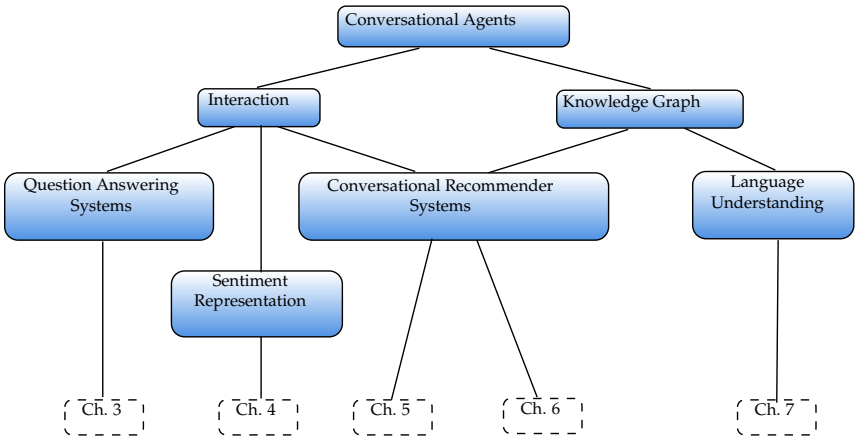


Figure 1.1: Thesis organization.

the chapters with the related topics of interest and their interconnections.

Chapters 4 to 7 share a standard structure. The sections follow a presentation flow that first introduces the research purposes and motivations of the specific study and then provides a comprehensive analysis of the related work. Thus, the composition of the proposed approach and associated experimental setting populate the remaining sections. Only Chapter 3 slightly differs from the others since its research focus is on reviewing the literature on interactive QA systems. As a further guide for readers, we collect all the acronyms that fill this thesis in Table 1, each with a dedicated explanation.

1.2 Research Contributions

The main objectives of this work are the analysis and new model proposal of the conversational system's interactivity and the exploration of several benefits brought by KGs when included in such NL systems. Although they differ significantly in the treated topics, their focus strictly interconnects with each other in the context of CAs. In particular, each part investigates a different perspective on dialogue systems, which equally conveys improvements to the main subject of this work. In the following, we dis-

Acronym	Explanation
ABSA	Aspect-based Sentiment Analysis
AC	Aspect Category
AI	Artificial Intelligence
AT	Aspect Term
CA	Conversational Agent
CB QA	Classifier-Based Question Answering
CoQA	Conversational Question Answering
cQA	Community Question Answering
CRS	Conversational Recommender System
FAQ	Frequently Asked Questions
HCI	Human Computer Interaction
IQA	Interactive Question Answering
IR	Information Retrieval
KB	Knowledge Base
KB QA	knowledge-Based Question Answering
KG	Knowledge Graphs
k-NN	k-Nearest Neighbor
LDA	Latent Dirichlet Allocation
LM	Language Model
LP	Link Prediction
ML	Machine Learning
MRC QA	Machine Reading Comprehension Question Answering
NL	Natural Language
NLI	Natural Language Interface
NLP	Natural Language Processing
NLU	Natural Language Understanding
QA	Question Answering
RS	Recommender System
SE	Search Enginges
VA	Virtual Assistant
VQA	Visual Question Answering

Table 1.1: Comprehensive list of acronyms employed in this thesis and their explaination.

cuss additional details on the research questions that led our study to the contributions outlined in each chapter. Note that Giovanni Maria Bian-

cofiore is the corresponding author of the scientific publications related to the research contributions presented in this thesis[1].

1.2.1 Interactive QA Systems

The great diffusion of VAs such as Amazon Alexa, Google Assistant, Microsoft Cortana, and Apple Siri plays a crucial role in the recent advancements of CAs [139]. One of the principal tasks these VAs perform is to answer questions in several fields: health, weather, news, and shopping [18], to cite a few of them. Accordingly, the *interaction* between humans and QASs is becoming more and more natural. Thus, Interactive QA (IQA) systems are a solution to several problems QA systems raise in satisfying user requests, like disambiguation issues and exploring correlated facts given an initial question.

This chapter offers a detailed overview of the IQA methods prevalent in current literature. The analysis focused on revealing the different levels of interaction implemented in such systems, which identifies a relevant component to build conversational agents. It begins by explaining the foundational principles of QA systems and defining new notations and taxonomies to combine all identified works inside a unified framework. The reviewed published work on interactive QA systems is then presented and examined in terms of its proposed methodology, evaluation approaches, and dataset/application domain. We also describe trends surrounding specific tasks and issues raised by the community, shedding light on scholars' future interests.

Contributions. We analyze QA systems from earlier perspectives by highlighting the different solutions proposed in the literature. We focused on work published over the last ten years to provide a detailed picture of IQA systems. In detail, we provide formal definitions of different QA systems and the primary characteristics that IQA systems should possess. We also present a unified architecture highlighting the primary components

[1]The authors of the publications are alphabetically ordered. The corresponding authors of publications are reported in the articles.

of IQA systems and operations while conducting an in-depth classification of state-of-the-art from the methodological and application perspectives. Then, we give detailed sights of the most frequently used datasets and adopted evaluation protocols and metrics organized by paradigms and objectives.

1.2.2 Interaction with Aspect-Based Sentiment Analysis

Is it possible to group aspect terms extracted through unsupervised approaches into coherent domain aspect categories relevant to an ABSA scenario? Can a sentiment-based text representation improve the performances of sentiment-related tasks?

Nowadays, more and more social networks and Web platforms allow users to share their opinions and tastes on items of different types. This information is beneficial for several purposes, such as providing personalized recommendation services or understanding thoughts and introspections conveyed through text. Consequently, also human-agent interaction receives a consistent boost in terms of naturalness and efficiency [40]. Sentiment Analysis provides helpful methods to analyze these textual opinions (e.g., reviews) from a global point of view. In case we want a more detailed representation of the opinion represented in a text, ABSA identifies a valuable option thanks to its fine-grained level of text analysis.

In this Chapter, we have designed a processing pipeline to extract aspects domain-related from text using an unsupervised approach. We formally define Aspect Terms (ATs), Aspect Categories (ACs), and Aspect-based sentiment embedding, which allows representing documents by aggregating sentiment scores for each AC they contain. We perform experimental evaluations on the Spotify dataset to prove the utility of our technique in predicting elements strictly related to emotions and feelings. Our results show improvements in the regression task for sentiment-related features compared to the classical semantic-based representations.

Contributions. We introduce an unsupervised method to extract relevant aspects from the text for Sentiment Analysis. Our proposed solu-

tion groups all the found ATs into ACs, each characterized by a specific sentiment value. To highlight these distinctions, we also provide formal definitions about *Aspect Term* and *Aspect Category*. Moreover, we present the concept of *Aspect-Based Sentiment embedding*, a representation strategy of textual documents based on domain aspects and their related sentiment scores. We demonstrate the utility of this embedding technique by designing an experiment that proves our approach grants improved performances in predicting textual features related to human feelings against the well-known semantic approaches.

1.2.3 Conversational Recommender

Can CRSs improve the efficacy and naturalness of the task employed in an existing RS service? May a mobile app improve the job recommendation task?

Information overload is a well-known problem that impacts users' digital experience when they need to find interesting items in a large set of possible options [98]. In the literature, IR and RS research areas investigated this problem. In the first case, implemented systems do not consider the user's past preferences and retrieve the most relevant items according to the user query. On the other hand, even though the goal of RSs and IR systems might seem similar, a user's profile is built, allowing the proposal of services tailored to the user's specific characteristics.

This Chapter outlines the recommender system proposed in *GUapp*, based on Latent Dirichlet Allocation (LDA) and the k-Nearest Neighbor (k-NN) to compute job positions most suitable to the user profile. Furthermore, to improve the user experience, *GUapp* implements a chatbot whose goal is to allow users to interact with the app through natural language. Thanks to that, the search and recommendation process becomes incremental, and the user can add new requirements at each stage of the interaction, achieving an effective recommendation through conversation.

Contributions. We developed *GUapp*'s RS, which classifies jobs according to their topics, and a k-NN method computes a personalized recommendation list. That list is updated daily and provided to the user

while keeping the user's past preferences to rank the job positions based on her interest. Moreover, *GUapp* offers a natural-language-based interaction through a chatbot. The *GUapp*'s chatbot allows the user to define her interests, describe her skills, and filter out results that do not match her requirements. The user interface is dynamically adapted to the user behavior to show the most fitted view as the app opens. Hence, it improves the user's overall experience by anticipating her needs, delivering the list of new jobs, the recommendations, or the chatbot view at startup, based on the user habits.

1.2.4 Knowledge Graphs for leading Dialogues

Does integrating a KG in a CRS help drive the dialogue, making the interaction with users more natural and pushed at a more refined-grained level?

RSs feel the significant issue of cold-start and the possible lack of items to suggest to users. With a CA which interacts with users exploiting a domain-specific KG, it is possible to solve those problems. Indeed, the main difference between a CRS and a standard RS is the interaction with the user that is more efficient, and natural [244]. Accordingly, a CRS lets the system build the user profile during the interaction, allowing her to express preferences through a human-like dialog, especially suited to Knowledge-based and KG-based RSs.

With this Chapter, we define an improved version of *GUapp*, an ecosystem for job-postings search and recommendation for the Italian public administration. Its main goal is to match user skills and requests with job positions on the Gazzetta Ufficiale website, offering recommendation services through conversations. A domain-specific KG empowers *Guapp*'s dialogues, which improves the users' natural language interaction with the app and their user experience. Thanks to that, the search and recommendation process becomes incremental, and the user can dynamically provide her preferences at each interaction stage.

Contributions. We present GUapp and its overall architecture, besides the functioning of the conversational agent that dialogues with the user by

exploiting a custom-designed Knowledge Graph. We also show a running example that outlines how GUapp models users and provides practical recommendations through natural language conversations.

1.2.5 Knowledge Graphs for Natural Language

Does BERT generate a latent semantic space holding information about Knowledge with explicit semantics? Can we learn functions to automatically detect precise knowledge graph concepts from the BERT latent space?

The advent of pretrained language has renovated the ways of handling natural languages, improving the quality of systems that rely on them. BERT played a crucial role in revolutionizing the Natural Language Processing (NLP) area. However, the deep learning framework it implements lacks interpretability [258]. Thus, recent research efforts aimed to explain what BERT learns from the text sources exploited to pre-train its linguistic model.

This chapter analyzes the latent vector space resulting from the BERT context-aware word embeddings. We focus on assessing whether regions of the BERT vector space hold an explicit meaning attributable to a KG. First, we prove the existence of explicitly meaningful areas through the Link Prediction (LP) task. Then, we demonstrate these regions are linked to explicit ontology concepts of a KG by learning classification patterns.

Contributions. To the best of our knowledge, our work is the first attempt at interpreting the BERT learned linguistic knowledge through a KG relying on its pretrained context-aware word embeddings. We prove that BERT embeddings already possess information about the structure of a KG without needing any fine-tuning processes or architectural changes. We demonstrate that inner spatial properties of the BERT vector space allow inferring explicit KG concepts, with extensive experiments supporting our findings.

1.3 Final Remarks

This dissertation collects all our efforts in reviewing and proposing models and approaches significant to the central topics of the thesis, which are CAs. However, a complete dialogue system intertwines solutions from several variegated fields, which flow into dedicated research topics. Thus, our investigations cover the primary objectives of the cases that characterize them, while as further goals, the reached findings enrich the proposals of this work.

For completeness, Chapter 2 will give a comprehensive panorama of the research interests portraying the fields mentioned above. In detail, we summarize all the major concepts that the reader has to employ to be aware of the topics addressed in this document, besides encompassing those intuitions that bring novelty through our proposals. Moreover, we outline metrics and datasets highly acknowledged by the community to guarantee a constructive and exhaustive reading of this thesis through its chapters.

2

Background

In this chapter, we will provide all the theoretical background, good practices, and accepted conventions unanimously from the researchers' community regarding the main topics treated in this dissertation. Such information, wisely collected and distributed among the following sections, will give the reader a basic knowledge of Conversational Agents, Interactivity, and Knowledge Graphs to deal with our research product. First, we present a frame of today's panorama about CAs with accepted definitions, taxonomies, and applications. Then, we give some necessary considerations about the interactivity of such systems, deepened in Chapter 3, and a description of KGs and their usage.

2.1 Conversational Agents

Conversational agents (CAs), also known as intelligent virtual agents, are computer programs designed for natural conversation with humans [236]. CAs may entertain informal chatting, in which case the system takes the name of a *chatbot*, or it enables the user to fulfill specific tasks (such as a flight reservation, information seeking, etc.), in which case the system is called a *task-oriented agent*. However, CAs state their fame thanks to the natural and intuitive way of interacting through speech [108] and time-

saving via hands-free possibilities [147]. Recent advances in the NLP and AI state-of-the-art allow empowering the functionalities of CAs such that users can also enjoy hedonic and social interaction. Nevertheless, there is still a need for improvements concerning the interactivity and cognitive abilities these systems offer to their audience, which is the purpose of this dissertation.

Definition 2.1.1. *Conversational agents* or *chatbots* are software-based dialogue systems designed to simulate a human conversational process by processing and generating natural language data through a text or voice interface to assist users in achieving a specific goal [138]. □

Depending on the goals a CA aims to achieve with its implementation, it may consistently vary in its design and categorization. Indeed, Wahde et al. [236] propose a coarse-grained taxonomy of CAs, classifying them into *chatbots* and *task-oriented agents*. The former are systems that maintain generic dialogues with users rather than provide precise information. The latter solves an opposite task: providing clear, relevant, and definitive answers to specific queries. Then the authors propose a set of different characteristics that each CAs belonging to those categories may have to define slightly different agents (i.e., input modalities and dialogue representations).

On the other hand, Hussain et al. [93] interchangeably use the terms CA and chatbot, simply distinguishing the taxonomy mentioned above into task-oriented and not-task-oriented agents. Instead, they articulate the CAs' categorization on a finer granularity level. This distinction considers four criteria:

- Interaction mode (e.g., textual or not);
- Goal (i.e., task-oriented or not);
- Design approach (e.g., rule-based or AI-based);
- Domain (i.e., closed and open domain).

The exact taxonomy guided the work of Motger et al. [138], although their survey presents a more detailed and meticulous description of the state-of-the-art regarding today's CAs. However, without leading the generality of this thesis, we offer in the following a summary of the primary CAs that populate the literature based on two criteria (i.e., domain and goal). We forward the reader to the works mentioned above for further insights beyond this Chapter's scope.

2.1.1 Domains

Chatbots widely differ in their behavior depending on the domains they engage users. In detail, CAs handling general-purpose objectives may interact more friendly with their audience in their language register and the politics of leading dialogues and computing answers. Instead, agents that treat domain-specific activities result in more rigorous functionalities, often requiring users' expertise in the related field. Based on solutions available in the literature, CAs occupy the following areas.

General Purpose. The most common domain in research is the support of daily life activities [12, 107], where CAs help users achieve their personal goals and activities. In addition to general-purpose chatbots like QA systems (cf. Chapter 3), this domain includes different domain-specific application areas like restaurants and food [105], smart-home assistance [63], and commercial activities like e-customer service support [70] and e-commerce assistance [233]. For the latter, the agents usually take the form of CoRSs or VAs (cf. Section 2.1.2).

Technical. Chatbots that match this category refer to complex technical infrastructure management and software engineering assistance. This domain covers end-user technical support [222] to advanced, semi-automatic management of smart industrial infrastructures [79], including telecommunication systems like autonomous call centers [155] and tools for bridging the gap in terms of usability between users and software products or devices [88].

Healthcare. It is a very active area of research, especially for some (non-

critical) tasks in elderly care. Examples include CAs that monitor medicine intake [59], interact with patients regarding their state of health (e.g., during cancer treatment [178]), or provide assistance and companionship. Also, CAs increasingly find application in treating mental health problems [229], generally with positive results. They can also be used in lifestyle coaching, discouraging harmful practices such as smoking [172] and promoting healthy habits and lifestyle choices involving, for example, diet or physical exercise [60].

Education. Most educational applications reported below rely on interpretable CAs, although black box CAs have also found an application. A natural application of CAs in education is to provide a scalable approach to knowledge dissemination. In such settings, they can improve the course quality by setting up handling requests from hundreds or thousands of students simultaneously [248]. Other examples of educational CAs provide learning support for visually-impaired adults [113] and deaf children [41].

2.1.2 Goals

The goal of CAs strongly affects their design and implementation, which is, in turn, influenced by the application domain. As we highlighted in Section 2.1.1, each field requires CAs to assume one to more behaviors. For instance, general-purpose chatbots can answer generic audience questions (QA systems) or filter tailored information by searching for interesting items (CoRSs). In contrast, healthcare CAs undertake the only scope of assisting users (VAs). To this aim, we can classify CAs depending on the following goals.

Information Seeking. CAs that identify commodity query engines allow easy-to-use access to knowledge or databases (either generic or domain-specific). Conversational agents for information requests mainly include QA chatbots [44], comprehensively defined in Definition 3.2.1.

Information Filtering. CAs that include daily use, domain-specific utilities like restaurant recommendations [105], and tourist-experience advisors [162]. The information is processed through information aggregation

algorithms to provide personalized answers based on auxiliary data (e.g., user information, context data). Such CAs have recently found an extensive interest in literature with the name of Conversational Recommender Systems (CoRS).

Definition 2.1.2. A CoRS is a software system that supports its users in achieving recommendation-related goals through a multi-turn dialogue [97]. □

Assistance. The most common goal among CAs is to extend software systems with a support tool by integrating human-like communication features into existing activities and processes to enhance user experience. Some application contexts are the commerce domain, as customer and service support tools [21]; the business domain, as semi-automated interfaces of utility processes and for internal support [42]; the healthcare domain, in the form of psychotherapy [4]. In few words, such systems vocated to the audience's assistance take the name of Virtual Digital Assistants.

Definition 2.1.3. A Virtual Digital Assistant is an application program that can understand natural language and complete (electronic) tasks for the end user [276]. □

2.1.3 Evaluations

The evaluation of CAs is very much an open topic of research. There are many evaluation criteria aimed at different levels of human-agent interaction. On a low level, when developing the capabilities of a CA to process utterances, retrieve information, and generate language, traditional metrics for general pattern recognition, like precision and recall, represent useful metrics for evaluation. For example, for a CA employed in a psychiatric facility, one may wish the agent (or a component thereof) to assess whether signs of depression transpire from the interaction with the patient: In such a case, maximizing recall is essential.

Traditional metrics like precision and recall are employed across several benchmark problems, e.g., to evaluate sentiment analysis [215], question

answering [196], and reading comprehension [271]. The general language understanding evaluation (GLUE) benchmark [238] provides a carefully-picked collection of data sets and tools for evaluating natural language processing systems and maintaining a leaderboard of top-scoring systems.

Alongside traditional scoring metrics, the field of natural language processing has produced additional metrics that are useful to evaluate CAs in that they focus on words and n-grams (sequences of words) and are thus more directly related to language. A relevant example of this is the bilingual evaluation understudy (BLEU) scoring metric. Although initially developed to evaluate machine translation quality [169], BLEU resulted relevant to assess many aspects of CAs, e.g., summarization and question answering. In short, BLEU can be seen as related to precision and is computed by considering how many n-grams in a candidate sentence (e.g., produced by a CA) match n-grams in reference sentences (e.g., high-quality sentences made by humans). In addition, BLEU includes a penalty term to penalize candidate sentences that are shorter than reference sentences.

The evaluation of the quality of interaction requires human judgment instead. We identify two principal methodologies: interviews and qualitative questionnaires.

Interviews. Qualitative interviews allow evaluators to get detailed feedback from the participants of an experiment with the conversational agent regarding its impact and the effects of user interaction with the agent. Such interviews usually follow two different patterns: focus group interviews, where participants might be grouped into small groups to facilitate discussion and the exchange and contrast of opinions and impressions [198]; and individual custom interviews, which are designed and carried on separately for each individual based on their experience with the agent and their personal user experience as in open-ended, semi-structured interviews [213]. The most frequent metric for measuring and analyzing the participants' responses is the Likert scale measure for objective questions, alongside discourse analysis on interview transcripts.

Questionnaires. Composed of a list of questions regarding users' interaction with the CAs, questionnaires aim to assess specific system features

that strongly affect their design. Goal-oriented questionnaires measure and monitor features or qualities on which the interaction with the agent has a significant impact. For instance, in the healthcare domain, questionnaires can be used to measure disease symptoms like depression or anxiety before and after interaction with a conversational agent focused on dealing with and treating these pathologies [66]. On the other hand, user satisfaction questionnaires cover usability quality topics, including emotional awareness, learning, and relevancy of content [64]. In qualitative interviews, using a 5/10-point Likert scale is the most frequent metric used in questionnaires.

2.2 Chatbot Core Topics

Significant tasks and procedures are now demanded of CAs, which undoubtedly facilitate several duties of people. In the previous sections, we have highlighted the prominence characterizing chatbots in domains, the goals they accomplish, and the evaluation methods designed to assess their quality. However, users still perceive such technology as a service provided to individuals with specific processes or social agents with their intentions depending on internal design [26]. Thus, there is a growing need to consider CAs trustworthy in human interaction.

To this aim, features like interactivity and performances became essential for the audience. The former allows users to relate with agents naturally and proactively so that the overall interaction benefits both actors (i.e., reaching objectives in a few conversational steps or feeling comfortable in any language setting). The latter further boosts previous profits, eventually permitting lighter system implementation on a broader range of applications. In detail, KGs represent a valuable resource in amplifying the CAs' performances in guiding users to reach their objectives rather than understanding their intents or generating valid natural language responses.

In the following sections, we provide some fundamental concepts (i.e., interactivity and Knowledge Graphs) shaping the intuition behind this research work, then deepen through the remaining Chapter.

2.2.1 Interactivity

The term interactivity presents numerous meanings across several domains. Generally, interactivity is the responsiveness one experiences from another entity independently from their nature. Although dealing with CAs allows research to model several protocols and practices to analyze such property objectively, it is well known that interactivity objectively varies based on differences in communicators' conversational behaviors, whether face-to-face or via other media [191]. Therefore, differences in interactivity may be associated with various significant interpersonal impressions and outcomes [120].

In this thesis, we principally refer to interactivity as the capability of a system to handle and solve natural language issues (e.g., ambiguity in words and user intents) by actively engaging users through dialogues or employing multimedia information. To this purpose, we dedicate Chapter 3 to formally frame this interaction, with a comprehensive focus on QA systems and their literature.

However, we know systems interactivity cannot be limited to being analyzed on the dimension of improving performances. Indeed, the user's impressions of the system interacting with it are crucial in defining its interactivity. CAs have many applications in the healthcare and customer service domains, so it is necessary to handle the conversation with some degree of empathy to be effective [29]. Thus, Sentiment Analysis provides practical techniques to leverage agents' interactivity in covering this other analysis sphere of human-computer interaction.

Definition 2.2.1. Sentiment Analysis (SA) or Opinion Mining (OM) is the computational study of people's opinions, attitudes, and emotions toward an entity [148]. □

In a nutshell, the SA aims to extract some subjective elements that do not directly leak out from the natural language text. Such a method allows further comprehension of user opinions and intents, granting CAs the ability to adapt to their audience state. In terms of interactivity, SA boosts chatbots to an additional level of involvement with the user. Most recent works

in the Sentiment Analysis field exploit supervised learning techniques that have proven effective for different text classification tasks. In particular, among different supervised approaches, there has been a lot of interest in Deep Learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNN) with all their variants (i.e., LSTMs and GRUs), or Language Models like ELMo [174] and BERT [51].

CNNs have shown powerful performances on the crucial task of sentiment classification [106, 104, 100]. The promising results of CNN-based models have driven researchers to further explore the Deep Learning field. For instance, in [77], authors proposed a joint CNN and LSTM framework that takes the features extracted by the CNN as input for the LSTM to perform sentiment analysis on short texts through a pre-trained word embedding model. Similarly, Behera et al. [15] investigated a hybrid approach of CNN and LSTM to analyze consumer reviews posted on social media, aiming to be domain independent and highly adaptable in examining big social data while keeping scalability. Recently, there has been a lot of interest in the combination of classical Deep Learning architectures and Attention mechanisms, whose goal is to mimic cognitive attention in finding relevant contextual elements in a sentence [216]. Many studies [32, 241, 13] have demonstrated that the combination of the previous techniques has improved performances on the text classification task. The advent of powerful language models like BERT defined a new trend in the NLP research field. In particular, there has been a lot of interest in training custom models for specific languages for solving different research problems, i.e., Sentiment Analysis [183, 179].

As opposed to extracting the general sentiment expressed in a text, Aspect-Based Sentiment Analysis (ABSA) aims to extract both entities described in the text (e.g., attributes or components of a product or service) and the writers' opinions about such entities. As a result, there has been an increased interest by the NLP community in ABSA and Aspect Term Extraction (ATE), so more resources have been made available for these tasks. Early works deal with the extraction of Aspect Terms and the related sentiment by using association rules [90], Conditional Random Fields

(CRF) [96], knowledge-based topic modeling [33], or double propagation [132, 133]. Recently, the research focus has moved to use Deep Learning models for ATE. For instance, in [182], authors did different experiments with Convolutional Neural Networks (CNNs) and several word embedding strategies. The main problem of CNNs is the handling of long-term dependencies. Thus, recent works started to adopt Recurrent architectures, such as LSTMs. The latter is used by Li et al. [128], where the history of aspects detection and opinion summary enhance the ATE model.

Other Deep Learning-based approaches also tried exploiting the dependency trees' information to enrich the sentence representations as input to the ATE models. Ye et al. [267] proposed a model based on CNNs, that uses Tree-Based Convolution to incorporate dependency relationships. Luo et al. [137] used a Bi-directional LSTM to learn a representation of the dependency tree for each review.

In the dedicated Chapter 4, the main focus is on extracting aspect terms and sentiment information from the raw text in an unsupervised way. In literature, different approaches implement unsupervised ABSA models. In [69], authors developed a framework able to perform both supervised and unsupervised ATE in large review datasets based on Bi-Directional LSTM networks and CRF. The idea is to automatically annotate a dataset in an unsupervised manner by only considering nouns and noun phrases as candidate aspects and then training the Deep Learning model in a supervised way. In [231], Vargas et al. proposed a simple approach called SUAEx for unsupervised aspect extraction, which relies solely on the similarity of word embeddings. He et al. [81] introduced a novel neural approach to discover coherent aspects exploiting the distribution of word co-occurrences through the use of neural word embeddings. In addition, they implemented an attention mechanism to de-emphasize irrelevant words during training, further improving the coherence of aspects. In [31], authors designed a two-step hybrid model by combining linguistic patterns with Deep Learning techniques to improve the ATE task. The first step exploits a rule-based approach to extract aspect terms, used as input for the second step, which consists of training attention-based Deep Learning model.

Other works focused on implementing domain-independent Deep Learning ATE models trained on multiple datasets (e.g., restaurant, laptops, and hotel reviews) to improve their robustness and generality [170].

2.2.2 Knowledge Graphs

Knowledge graphs use a graph-based data model to capture knowledge in application scenarios that involve integrating, managing, and extracting value from diverse data sources at large scale [164]. This data structure allows facts and concepts to have a formal and strict shape so that systems can have direct access to generic human knowledge easily and automatically. Indeed, graphs provide a concise and intuitive abstraction for various domains, where edges and paths capture different, potentially complex relations between the entities of a domain [8]. Moreover, KGs permit the emulation of actions typical of human thought, like the definition of semantics, entailments, and reasoning.

Definition 2.2.2. A knowledge graph is a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities [85]. □

Formally, given E, R, and F sets of entities, relations, and facts, respectively, a Knowledge Graph is defined as $G = \{E, R, F\}$, where $(h, r, t) \in F$ is a triple of subject h, predicate r, and object t. Thus, $h, t \in E$, also called respectively head and tail entities, and $r \in R$.

2.2.3 Conversational Recommender

This section cuts across two principal research areas: recommender systems and conversational agents. Recommender Systems (RSs) can, in turn, distinguish two main classes according to the paradigm implemented: Collaborative Filtering (CF) and Content-based (CB). Independently from the paradigm adopted, any RS aims to help users address the information overload problem. RSs prioritize the delivery of information for individual

users based on their learned preferences [119]. Hence, RSs support the user during decision-making when she has to decide among a large set of options. In CF, the recommender system exploits the user community to identify items potentially interesting for a given individual. This kind of RS is generally implemented in online platforms like Amazon [214], Netflix [72], and so on. The basic idea behind this recommendation strategy is that similar users like similar items. Conversely, CB RSs do not need any community, and recommendations rely on matching user preferences and textual content associated with the items [98]. The specific task of job recommendation is widely investigated in the literature given the significant diffusion of internet-based recruiting platforms [166]. In the past, the most used approaches for job recommendation foreseen a boolean search and filtering [142]. Later, efforts moved on to the problem of catching user preferences and building a user profile. In [193], the authors propose a system that builds the user profile by passively detecting users' click-stream and read-time behavior. It is the same strategy adopted in `GUapp` to reduce the user effort for defining her preferences. Malinowski et al. [142] build a multi-slot profile with information about demographic data, job experience, language, and IT skills. As regards the recommendation algorithm, Amato et al. [7] compared several algorithms for matching candidate profiles with job descriptions. From their study, LDA emerged as the best algorithm in terms of precision, recall, and f-measure, compared to other machine-learning and rule-based approaches. LDA found an application in [11] for building a set of job postings and user profile features. [48] proposed a hybrid approach. The authors combine content-based and KNN in a fashion similar to that used in `GUapp`.

The second relevant research area for this work is Conversational Agents (CAs). CAs are software agents able to interact with the user through natural language. Generally, there are two main classes of CAs: end-to-end and modular systems [10]. The former typically use neural networks [207, 235, 217, 55] and learn a dialog model from a set of past conversations [24]. These systems demonstrated good performances in chit-chat dialogs. The modular systems are pipeline-based agents and are composed of modules,

each with a specific function [56, 274, 247, 131]. Modular agents generally work fine for the goal-oriented system. A Conversational Recommender System (CoRS) is a goal-oriented system whose goal is to make recommendations in a given domain. The main difference between a CoRS and a traditional RS is that the interaction with the user is more efficient and natural [244]. Accordingly, the construction of the user profile is incremental, and the user can express her preferences at different stages of the recommendation process by an interactive human-like dialog [102]. That is the aspect that we tried to emphasize by implementing a chatbot in `GUapp`. There needs to be more extensive literature on CoRSs. In [192], the authors propose the idea of combining Virtual Assistants and CoRSs. Recently, CoRSs have been effectively used and tested in different domains such as music, movie, and book [160, 94].

The task above is well suited to Knowledge-based and KG-based RSs. An example is [78], where a comprehensive KG is built upon a specific domain to lead dialogues and recommendations.

3

Interactive QA Systems

QA systems are a popular and frequently effective means of information seeking on the web. In such systems, information seekers can receive a concise response to their queries by presenting their questions in natural language. Interactive QA (IQA) is a recently proposed and increasingly popular solution that resides at the intersection of *question answering* and *dialogue systems*. On the one hand, the user can ask questions in everyday language and locate the actual response to her inquiry. On the other hand, the system can prolong the QA session into a dialogue if there are multiple probable replies, very few, or ambiguities in the initial request. By permitting the user to ask more questions, interactive question answering enables users to interact with the system and receive more precise results dynamically.

This Chapter explains the foundational principles of QA systems and defines new notations and taxonomies to combine all identified works inside a unified framework. The reviewed published work on IQA systems is then presented and examined in terms of its proposed methodology, evaluation approaches, and dataset/application domain. We also describe trends surrounding specific tasks and issues raised by the community. Our work receives further support from a GitHub[1] page hosting a synthesis of

[1]https://sisinflab.github.io/interactive-question-answering-systems-survey/

all the major topics covered in our investigation.

3.1 Introduction

Researchers have usually contrasted QA systems with Search Engines (SE) in the early literature. The primary distinction was that the latter returns a ranked (often lengthy) list of documents, whereas the former produces an *answer* to the question posed by the user [84]. Today, the boundary between SEs and QA systems is becoming increasingly blurred [58]. When we query Google seeking for "President of the United States of America", we no longer receive a list of results in which we must search for the answer, but rather a concise snippet that contains the answer to our question (extracted from a web page).

Essentially, QA systems share the characteristic of providing a clear answer to the user inquiry, regardless of the type of question, i.e., factual ("*Which kingdom does the animal Bird of Paradise belong to?*") [130], visual ("*What color hair does the woman have?*" combined with a picture of a woman) [208], or open-goal oriented ("*How can I connect my Fitbit sensors to the server?*") [76]. The answer can be extracted from a portion of a document (as in the preceding example) or generated by condensing/summarizing fragments of many words into a coherent whole [84]. *Extraction* and *generation* constitute the core blocks of a QA algorithm.

The QA task progresses from the earliest systems characterized by one-shot requests through IQA, which allows users to continue interacting with the system after receiving an answer. In some recent conversational question answering, a multi-turn dialog provides for determining the correct answer. In this direction, the great diffusion of VAs such as Amazon Alexa, Google Assistant, or Apple Siri is playing a crucial role [139]. One of their principal tasks is to answer questions in several fields: health, weather, news, shopping [18], to cite a few of them. Accordingly, the *interaction* between humans and QA systems is becoming more and more natural. Here, we use the term *interaction* to refer to any feedback or reaction/response from the user or system, not only the standard exchange of questions and

answers seen in Conversational QA (CoQA) systems.

Motivations. IQA systems are a solution to several problems raised by QA systems in satisfying user requests. One of the core problems is related to the *disambiguation* of the user question. The user request is not always understandable by the system, and part of the interaction can be devoted to the disambiguation or, more generally, clarification of the user request. This scenario does not affect only conversational systems but interactive ones in general. If we query Google with *"When was Milan founded?"*, the answer provided is related to the *"Milan football club"*. However, the related questions proposed by Google also contain *"When the city of Milan was founded?"*, since *Milan* is an ambiguous name. In case the user did not mean *Milan* as the football club, but as the city, she can click on the related question and get the right answer.

Thus, the watershed between *non-interactive* and *interactive* QA systems will be the capability of the latter to continue the interaction after the original query for *disambiguation* or *exploration* purposes. About the exploration, in a system-driven scenario, the system proposes a set of other questions related to the topic (e.g., Milan, football clubs, etc.), and the user selects one of them. Conversely, in a user-driven scenario, the user can ask further questions after receiving answers.

Undoubtedly, the latter is the most challenging, as highlighted by Guo et al. [74], between IQA and CoQA systems. In addition to the general problems of an IQA task, there are some particular issues related to natural language processing. For example, these systems should be able to cope with complex linguistic figures such as the ellipsis phenomena (e.g., *"Where was the President of the United States born?"*, *"Where did **he** graduated from?*) that characterizes dialogues between human beings. For this reason, a CoQA system has to keep track of the interaction *state*, introducing an additional analysis dimension, exploited in this survey, related to the *stateful* or *stateless* characteristic of these systems.

Contributions. We analyze QA systems from the aforementioned perspectives by highlighting the different solutions proposed in the literature. We focused on work published over the last ten years to provide a detailed

picture of IQA systems, such that:

- We provide formal definitions describing the purposes of QA systems in the literature and further state the primary characteristics that IQA systems should possess;
- We present a unified and encompassing architecture highlighting the primary components of IQA systems and their operation;
- We lead an in-depth classification of state-of-the-art from the methodological and application perspectives;
- We give detailed sights of the most frequently used datasets, adopted evaluation protocols and metrics, organized by the paradigms and objectives of IQA systems;

3.2 Question Answering Approaches

The focus is on IQA, which is a multidisciplinary topic combining several fields such as Information Retrieval (IR), NLP, HCI, and, more recently, AI and ML. Thanks to the latest developments in NLP and ML areas, QA systems have found a significant and growing interest from the AI community, especially for the role they play for digital assistants (e.g., Google Assistant, Amazon Alexa, Apple Siri). Nowadays, they have evolved to gain their field of study, whose goals continually increase by including new research topics like knowledge representation and semantic entailment.

Thus, we present the formal definition of IQA, including the QA problem (cf. Definition 3.2.1), two configurations of IQA (cf. Definition 3.2.2, Definition 3.2.3) and CoQA systems (cf. Definition 3.2.7). Then, we provide definitions of interactive session (cf. Definition 3.2.4), QA state (cf. Definition 3.2.5) and conversational history (cf. Definition 3.2.6), and different examples/pointers to state-of-the-art solutions that would help the reader to obtain a profound understanding of the topic.

Assumption. In information-access systems, the user's information need could be expressed through *keywords* (retrieval), a *question* (question-

answering), or *user profile* (recommender system). At the same time, answers could be a piece of text, an image, or items of interest most relevant to the information needed. Systems such as task-oriented chatbots or dialog systems whose primary role is not providing information access remain outside the focus of this survey.

To highlight the motivations behind this assumption, we consider the following definition.

Definition 3.2.1 (QA problem)**.** Let $Q = \{q_1, ..., q_N\}$ be the set of possible queries and $A = \{a_1, ..., a_M\} \cup \{\texttt{NULL}\}$ the set of possible answers including the `NULL` symbol representing the situation where the system is not able to provide an answer. A QA system aims to find the most relevant answer to a question in a single shot iteration. More formally:

$$\forall q \in Q,\ \hat{a} = \underset{a \in A}{\operatorname{argmax}}\, g(q, a)$$

with g being a utility function that considers how well a given answer a satisfies the proposed query. In probabilistic terms, we can define the problem as finding the most probable answer given an input question (and its context)

$$\forall q \in Q,\ \hat{a} = \underset{a \in A}{\operatorname{argmax}}\, p(a|q)$$

□

Users may interact with QA systems to find the information they need. A QA system will answer the question with a unique result well-formed in natural language, like *"The Lord of the Rings writer is J.R.R. Tolkien"*, saving the user to search for the needed information.

3.2.1 Interactive QA as Exploration and Disambiguation

Finding the correct answer to a given question is the primary goal to be achieved by QA systems. Although most of the QA systems show high performance for this task, some intrinsic natural language issues cannot be solved by only analyzing the submitted question. The definition of a

well-disambiguated request through a natural language question is a challenging task. Indeed, it requires the usage of specific terms plus a cognitive effort that is only affordable for some. Hazrinaa et al. [80] examine the semantic QA task, which permits the disambiguation of queries by leveraging context or requesting further information from the user.

An extension of QA systems, named **Interactive Question-Answering** systems, overcome this limitation with two specific goals:

Disambiguation. In case there are *too many* or *too few* eligible answers for a given question, or the request is ambiguous, the system can ask a new question to the user [110]. For this reason, IQA systems can be seen halfway between *QA* and *dialog systems*. Let us consider the case when the original query q has a set $A' \subseteq A$ of candidate answers. The system can suggest a new query $q' \in Q$. The user answer a' to q' gives the information that leads to an unambiguous answer $\hat{a} \in A'$ to the previous question q. We can say that $\hat{a}$ is an answer to the combination of q and q'.

Exploration. After the system returns the answer $\hat{a}$ to the initial query q, a new set of queries Q' can be suggested by the system or posed by the user[2] to explore other relevant topics related to $\hat{a}$.

As noted, interactivity refers to the possibility of the system posing or suggesting new queries to the user. Please note that while the disambiguation step is always **system driven**, exploration can also be **user driven**. In the latter case, the user decides the new aspects to explore related to $\hat{a}$.

Disambiguation and exploration may run for one step only or a sequence of steps. Depending on the "memory" the QA system has about previous questions and answers, we may have a **stateless** QA system or a **stateful** one. This latter situation leads to what we call **Conversational Question-Answering** system.

In a conversation aiming to disambiguate the original query q, we may have situations where the answer a' to q' does not lead to $\hat{a}$. In these cases,

[2] Q' is also referred to as *follow-up questions* in the literature.

the system computes a new query q'' based on a' until it reaches $\hat{a}$. It is not helpful to have exploratory steps while trying to disambiguate q to compute $\hat{a}$.

Example 3.2.1 (IQA for disambiguation)**.** Let us consider the case in which a user needs some information about music tracks, so she asks *"Who sang the song Money"*. This question is ambiguous since the answer could refer to the group *Pink Floyd* and the musician *David Gilmour*. In this case, the system replies with a question trying to disambiguate the user information needed. It searches for relevant information from the possible answers to reach this goal. Consequently, it will pose disambiguation questions until it reaches what the user meant. In this example, the system asks *"Do you mean the band who sang Money?"* and the user agrees, receiving then the final answer *"Pink Floyd"*.

It is worth noting that in Example 3.2.1, the choice of which disambiguating question to pose the user merely depends on the implementation of the system. Asking for *"Do you mean the musician who sang Money?"* leads to the same answer *Pink Floyd* once the user gives her feedback (i.e., she disagrees). That is because Example 3.2.1 has a binary mutually exclusive ambiguity. Choosing disambiguating questions from an often large set depends on whether the system allows optimized interactions.

Differently from disambiguation, two scenarios are possible in an exploratory conversation, where the QA system (system-driven) or the user (user-driven) may compute the queries. In a system-driven strategy, the new proposed questions have their answer already known. Thus, disambiguation steps become useless.

Example 3.2.2 (IQA for system-driven exploration)**.** Following the Example 3.2.1, a QAS could start an exploratory session within the conversation once it gives the needed answer. It could propose questions like *"Do you know when Pink Floyd was born?"* or *"Do you know when David Gilmour joined Pink Floyd?"* and then provide the related data depending on the user's answers. For instance, the user may know when the band was born, but she

does not know the answer to the second question, so the system will reply: *"He joined Pink Floyd in 1968 as a support to Syd Barrett"*.

Conversely, in a user-driven scenario, the user poses new queries, which may lead to ambiguous answers. The system must start disambiguation steps in those cases to reach the relevant solution.

Example 3.2.3 (IQA for user driven exploration)**.** Going back to the conversation in Example 3.2.2, the user may continue the dialogue by asking: *"When was he born?"*. Here we can refer to both *David Gilmour* and *Syd Barret*, so the system will ask *"Do you mean Syd Barret?"*. Consequently, the user may agree with the system or not. In the latter case, the QAS answers with *"David Gilmour was born on 6 March 1946"*. The conversation will last until the user information need is satisfied.

With respect to *stateful* exploratory QAS it results useful to avoid interaction loops. In principle, the exploration of the knowledge space may run forever. If we do not consider previous user interactions and answers, the exploration may get stuck in a loop where the system computes questions already suggested in the earlier steps.

Before we formally define the different interaction steps we have discussed, we introduce two unary operators to represent the possible actions an IQAS or a user may perform, e.g., generating a new query or an answer. We use $y \in \{u, s\}$ to state if the next query/ answer has been generated by the user (u) or by the system (s). Given $x \in A \cup Q$, $y \in \{u, s\}$, $a \in A$ and $q \in Q$ we will use:

- $x \overset{y}{\rightsquigarrow} q$: the IQAS ($y = s$) or the user ($y = u$) generates a new query.
- $x \overset{y}{\rightarrow} a$: the IQAS ($y = s$) or the user ($y = u$) generates an answer.

Given a query q, we then represent the simple query-answering step as $q \overset{s}{\rightarrow} \hat{a}$. Analogously, for the exploratory IQA we have $q \overset{s}{\rightarrow} \hat{a} \overset{s}{\rightsquigarrow} q'$ while the disambiguating Interactive QA steps are represented with $q \overset{s}{\rightsquigarrow} q' \overset{u}{\rightarrow} a' \overset{s}{\rightarrow} \hat{a}$. In this case, a' is the answer (e.g., feedback) provided by the user to the query q' generated by the system.

Definition 3.2.2 (Interactive Question Answering for Disambiguation)**.** An interactive question answering for disambiguation $\mathbf{IQA^d}$ is a system that takes a user query q as input and computes a new query q' to reach the answer $\hat{a}$ to q. More formally, we have

$$\hat{a} = \underset{a \in A, q' \in Q}{\operatorname{argmax}} p\left(q'|q\right), p\left(a|q \overset{s}{\rightsquigarrow} q' \xrightarrow{u} a'\right)$$

□

The idea here is to find the best next question q' given the initial query q in order to compute the best answer to $q \overset{s}{\rightsquigarrow} q' \xrightarrow{u} a$. As for $p\left(q'|q\right)$ and $p\left(a|q \overset{s}{\rightsquigarrow} q' \xrightarrow{u} a'\right)$, in principle, we do not make any assumption on their (in)dependency.

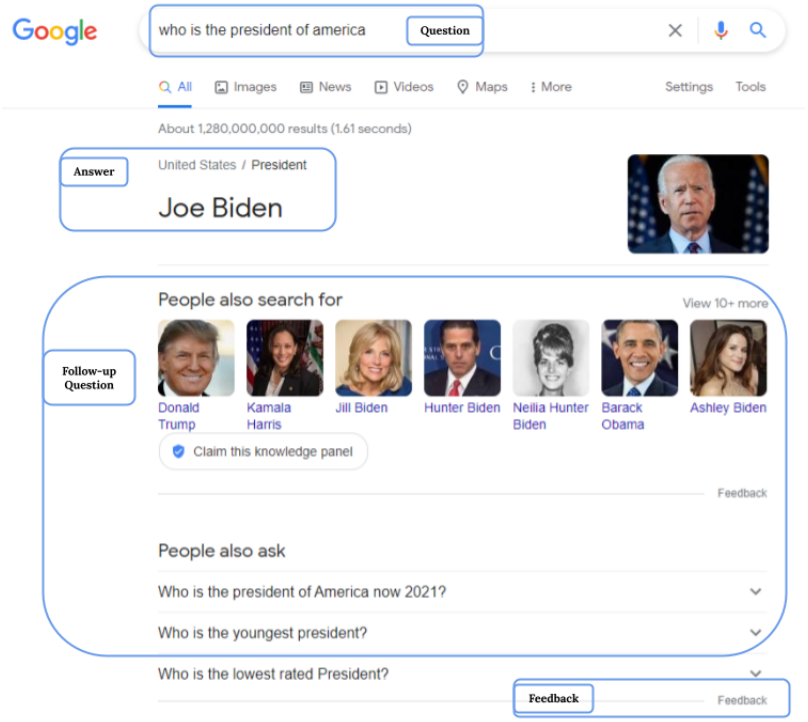


Figure 3.1: Example of an interactive QAs answer.

Figure 3.1 depicts an example of IQA system. The question asked to

the interactive system is *"Who is the president of America?"*. As already highlighted in the example, the user obtains an answer that satisfies her query, plus some more suggestions useful to explore the related exploration space. Here the picture outlines a collection of information and interactive area that allow users to continue their exploratory interaction.

The first information shown by Google in the example is the answer *"Joe Biden"*, which includes the entities and relations of a knowledge graph recognized in the question, respectively *"United States"* and *"President"*. The second one proposes a set of follow-up queries that could come after the initial question. For instance, other people searched for *"Who was the previous president of America?"* that was *"Donald Trump"*, or *"Who is the vice-president of America?"* that is *"Kamala Harris"*. In this way, the user may reach all this data by simply interacting with the suggested images, enriching the solution to her information need. The same area also groups different clickable questions people ask to achieve the same correlated answers. In a way, it gives information about the context understood by the system. The user who obtains an unexpected answer could rephrase her question to disambiguate its sense intention/meaning and then receive the appropriate response from the system.

Furthermore, the system provides a suite of possible multi-modal interactions, like images, audio, and text, which aim at improving both the user experience and the correctness of the results. A feedback slot allows people to interact further with the system by offering comments about the answer. Whether the solution is wrong, the user can specify the motivations and send them to the system, improving its behavior about that type of question.

Definition 3.2.3 (Interactive Query Answering for Exploration)**.** An interactive question-answering system for exploration **IQA$^{\mathbf{e}}$** aims at guiding the user through the exploration of a knowledge space. Given a question q, it provides both an answer and a set of follow-up questions close to q. More

formally, we have:

$$\langle \hat{a}, \hat{q}' \rangle = \underset{q' \in Q, a \in A}{\operatorname{argmax}} p\ (a|q), p\ (q'|a)$$

□

The aim of an IQA^e is to find an answer to q and, simultaneously, to suggest a new query q' related to the computed answer. Without loss of generality, in the previous definition, we consider only one follow-up question q', which can also refer to a set Q' of follow-up questions. Also in this case, we do not make any assumption on the (in)dependency between $p(q'|a)$ and $p(a|q)$.

Therefore, we may indeed have a hybrid situation where a disambiguation step supports the exploration:

$$q \overset{s}{\rightsquigarrow} q' \overset{u}{\rightarrow} a' \overset{s}{\rightarrow} \hat{a} \overset{s}{\rightsquigarrow} q''$$

Where the system must disambiguate the initial query before computing the answer $\hat{a}$ and the next question q''. An IQA system allows the user to further interact with the system once it replies through **interactive sessions**.

Definition 3.2.4 (Interactive Session)**.** A set of user-enabled actions following the system reply to a previous user question defines an *interactive session*:

$$I = \{a', q' : a \overset{u}{\rightsquigarrow} q', q \overset{u}{\rightarrow} a'\}$$

□

An *interactive session* with infinite cardinality remarks an interaction led by the user, which reflects her freedom to pose any question to the system response. In contrast, a finite *interactive session* cardinality outlines interactions conducted by the system. Therefore, the IQA system proposes only a fixed number of actions permissible (i.e., answers to disambiguating questions or enabled exploratory questions) to the user. Here, the interaction

occurs when the user gives her feedback to the *interactive session* (e.g., answering with disambiguating information).

3.2.2 Conversational QA

The interactive system shown in Figure 3.1 does not keep any memory of previous interactions with the user. When the user explores the knowledge space, every exploration step does not consider previous ones. However, the system needs a set of information to understand the user *intents* behind each question. With **context**, we refer to that data helping the system understand the meaning intended by the user behind her questions. Thus, the context is the set of data C which supports the QA system in finding the answer a to the user question q. Nevertheless, the overall process is a *stateless* application of a sequence of IQA^e. Accordingly, we can formalize the *state* as follows:

Definition 3.2.5 (QA State)**.** Let q be the user question, C a *context* supporting q, a the system answer and I the following *interactive session* enabled to the user. The *QA state* is the tuple that collects all the information of a QA interaction:

$$S_{QA} =< q, C, a, I >$$

□

In other words, the *QA state* hosts all the data exchanged through a user-system interaction, totally outlined at the end of each interaction.

The QA state context C determines the *statefulness* property of an IQA system. *Stateless* IQA systems host supporting information in their *QA state context* C that does not belong to other *QA states*, which may lead to loops in the exploratory task. On the other hand, if the system had a memory of previous QA states through the *context*, it could avoid proposing questions to the user already answered. Analogously, an IQA^d can only consider the original question q and the answer given by the user to q'. In general, IQA^d and IQA^e are *stateless* and do not take into account the **history** referring to previous interactions between the user and the system.

Suppose the IQA model had access to the user's search history. In that case, it could grant new rounds of interaction helpful, e.g., in handling some linguistic issues like the *Coreference Resolution* problem. The *coreference resolution* is the task of finding all expressions that refer to the same entity in a text [37], which frequently appears by exchanging follow-up questions with the system. For example, assuming that the first user query is *"Who wrote the Lord of Rings?"*, a follow-up question could be *"When **he** wrote **it**?"*, where *"he"* refers to the answer *"J.R.R. Tolkien"* and *"it"* to the entity *"Lord of the Rings"*. In these situations, the system must know what the user asked in the past and the given answers to solve this issue, thus moving from a *stateless* configuration to a *stateful* one. The *stateful* configuration of an IQA system is also known in the literature as *conversational* QA system due to its ability to treat conversations as having dialogues with the user.

In CoQA systems, we can have both user-driven and system-driven exploratory interactions and their combination. The previous example related to *"The Lord of the Rings"* is a clear user-driven exploratory interaction. Indeed, thanks to the *coreference resolution* and the corresponding *conversational context* (c.f. Section 3.3) computed by the CoQA system, the user is allowed to explore the knowledge space related to their original query q. Moreover, a CoQA system can trigger a system-driven disambiguation in case the answer to q' is not satisfactory to provide an answer to q. Thus, we will never need any disambiguation step in a *system driven* CoQA system for *exploration*. The new queries suggested by the system already have the answers (cf. Example 3.2.2). In the case of *user-driven exploration* via a CoQA system, a new query posed by the user after some exploration steps may require a disambiguation process by the system (cf. Example 3.2.3).

In a **System Driven Conversational Question Answering System for Exploration**, we have a sequence of exploratory steps in the form:

$$q \xrightarrow{s} a \overset{s}{\rightsquigarrow} q' \xrightarrow{s} a' \overset{s}{\rightsquigarrow} \ldots \xrightarrow{s} a^n$$

where the transitions between the *exploratory* questions posed by the sys-

tem and the related answers are interleaved by user feedback, in case, she is knowledgeable or not on those topics.

Dually, in a **User Driven Conversational Question Answering System for Exploration**, we obtain:

$$q \xrightarrow{s} a \overset{u}{\rightsquigarrow} q' \xrightarrow{s} a' \overset{u}{\rightsquigarrow} \ldots \xrightarrow{s} a^n$$

Differently from IQA^e, in the *conversational* case the system-generated query q^i at the i-th step also considers the two sets Q^i and A^i containing the previously generated queries $Q^i = \{q', \ldots, q^{i-1}\}$ as well as their corresponding answers $A^i = \{a, a', \ldots, a^{i-1}\}$. The same happens in the disambiguation case for the system-generated answer a^i.

Definition 3.2.6 (Conversation History and Conversation Span)**.** Given a conversational sequence of exploratory or disambiguation steps we define **Conversation History** at step i as $h^i = Q^i \cup A^i$ where $Q^i = \{q', \ldots, q^{i-1}\}$ and $A^i = \{a, a', \ldots, a^{i-1}\}$. For a step l and a step i, with $l < i$ we define **Conversation Span** as $s^{l,i} = h^i - h^l$.

A conversation history contains all the user-driven and system-driven interactions up to a certain point in the conversation, which feeds the context for the following question to answer. A conversation span represents the conversation history from a certain step l up to step i. We can see that $h^i = s^{0,i}$. So, what we will say for $s^{l,i}$ also holds for h^i. Again, the system generates the answer a^i to the query q^i and the following question to ask q^{i+1} in a system-driven conversational exploration. Also, since the system generates the following query, it is always unambiguous. Conversely, a user-driven conversational exploration could be different. The user-generated question q^i may be ambiguous to the system and require further interaction for disambiguation. Here, we can generalize concerning the definition of a IQA^d and assume multiple steps of interaction to disambiguate q^i in a situation like the following:

$$q \xrightarrow{s} a \underbrace{\overset{u}{\rightsquigarrow} q' \xrightarrow{s} a' \overset{u}{\rightsquigarrow} \dots q^i}_{exploration} \underbrace{\overset{s}{\rightsquigarrow} \tilde{q}' \xrightarrow{u} \tilde{a}' \overset{s}{\rightsquigarrow} \tilde{q}'' \dots \tilde{q}^k \xrightarrow{u} \tilde{a}^k}_{disambiguation} \underbrace{\xrightarrow{s} a^i \overset{u}{\rightsquigarrow} q^{i+1} \xrightarrow{s} \dots a^n}_{exploration}$$

where the conversational system cannot disambiguate q^i given a conversational span $s^{l,i}$. Then, it starts k rounds of disambiguation steps until it gets enough information to compute the answer a^i. Then, the user-driven interaction may continue to explore the addressed information space.

For the sake of presentation, we assume the system does not have a bounded number m of queries to pose during the disambiguation phase. In case, it can ask at most m questions to disambiguate. After the m-th answer cannot compute an unambiguous one, the system returns `NULL`, and the overall conversation ends.

Definition 3.2.7 (Conversational Question Answering)**.** A Conversational Question Answering system aims at exploring an information space alternatively via user-driven or system-driven interactions.

A **system-driven Conversational Question Answering system** $CoQA^s$ computes answers to the current query and the next question to ask, given a Conversation Span. More formally, we have:

$$\langle \hat{a}, \hat{q}^{i+1} \rangle = \underset{q^i \in Q, a^i \in A}{\operatorname{argmax}} p\left(q^{i+1} | a^i, s^{l,i+i}\right), p\left(a^i | q^i, s^{l,i}\right)$$

In a **user driven Conversational Question Answering system** $CoQA^u$, we formally distinguish between the exploration and the disambiguation as:

$$\hat{a} = \underset{a^i \in A}{\operatorname{argmax}} p\left(a^i | q^i, s^{l,i}\right) \quad \text{and} \quad \hat{q} = \underset{q^i \in Q}{\operatorname{argmax}} p\left(q^i | s^{l,i}\right)$$

□

Please note that in $CoQA^s$, the aim is to maximize the probability that a specific answer satisfies the current query and the likelihood of the following question. Conversely, in $CoQA^u$, we are only interested in computing the solution maximizing the probability that it satisfies the current query

during exploration. On the other side, in a disambiguation step, we are only interested in computing the following question to pose to the user to provide an answer.

3.2.3 Task and Challenge based classification of QA Systems

QA systems have several implementations to fulfill specific tasks, which decline the QA problem (cf. Definition 3.2.1) in multiple variations. It emerges from literature the following *task-based* taxonomy of QA systems.

Open-Goal QA. Here the QA systems exploit unstructured text to solve the QA problem. Forum messages or bounded sets of answers related to a specific domain, commonly known as Frequently Asked Questions (FAQ), fed the knowledge source of *open-goal* QASs. More in-depth, depending on the QA model and its knowledge source, the open-goal QA is further classified into *community* QA (cQA) [254] and *classifier-based* QA (CB QA) [136]. Selecting the best answer from a pool of candidate ones, usually built on top of a forum thread, is referred to as a *community* Question-Answering. In detail, cQA models foresee ranking mechanisms to achieve their goal. Conversely, a *classifier-based* QA system chooses appropriate answers by categorizing questions into default classes provided by the knowledge source (e.g., FAQ). Each group maps a specific answer that best satisfies the corresponding information needed.

Factoid QA. It answers questions that refer to a specific fact, e.g., *"Who is Leonardo Di Caprio?"* or *"What is Interstellar?"* and *"Where Christopher Nolan is born?"*. The QA knowledge source, which takes the shape of a *knowledge base* (KB), usually holds facts answering a given natural language question. The KB can occur as a set of unstructured documents hosting facts (i.e., Wikipedia, business documents, etc.) or as a collection of structured rules expressed in several forms (e.g., logic programming rules with Prolog and graph triples for KG). In case, we distinguish *machine reading comprehension* QA (MRC QA) [268] from

knowledge-based QA (KB QA) [275] tasks respectively. Teaching machines to read and understand texts on which to infer answers to user questions defines the *machine reading comprehension (MRC)* Question-Answering task. MRC QA systems reply to questions by pointing to words spanned in documents or by generating a new text string, enclosing facts satisfying the user information need. Differently, KB QAS implements a model to translate user questions into queries the KB allows for seeking answers. In other words, it provides a universally accessible natural language interface to factual knowledge [218].

Visual QA. This task aims to generate answers that encapsulate a truthful description of a picture on which questions emerge. Visual QA (VQA) seeks a correct answer for a given question consistent with the visual content of a given image [54].

	Open-Goal QA		Factoid QA		Visual QA
	cQA	**CB QA**	**MRC QA**	**KB QA**	**VQA**
SP	Wu et al. [254] Hu et al. [89] Wu et al. [253] Xiong et al. [260]	Waltinger et al. [237] Nie et al. [163] Su et al. [221]	Han et al. [76], Yang et al. [263] Yuan et al. [268], Chada [30] Das et al. [47], Mass et al. [145] Li et al. [123], McCarley [146] Xie [259], Li et al. [127] Bhattacharjee et al. [20] Kuo et al. [115], Kundu et al. [114] Osama et al. [165], Qu et al. [188] Wang et al. [242], Su et al. [220] Qi et al. [185], Li et al. [124] Yang et al. [264], Qu et al. [186] Qu et al. [187], Ju et al. [101] Zhu et al. [278]	Zheng et al. [275], Zhang et al. [272] Zhang et al. [269] Petukhova et al. [177] Perera et al. [171] Moon et al. [154] Damljanovic et al. [46] Liu et al. [130] Christmann et al. [36] Müller et al. [156] Shen et al. [209] Guo et al. [74]	Shi et al. [210] Do et al. [54] Gao et al. [67] Shao et al. [208] Pradhan et al. [184] Gordon et al. [73] Yang et al. [265]
UTTP	Rücklé et al. [201] Zhang et al. [273] Zhou et al. [277] Xiong et al. [260] Wong et al. [251]	Liu et al. [135] Latcinnik et al. [116] Sugiyama et al. [223]	Chiang et al. [34] Baheti et al. [9], Mandya et al. [143] Mandya et al. [144] Vakulenko et al. [230] Reddy et al. [197], Basu [14]	Sorokin et al. [218] Wu et al. [256] Habibi et al. [75] Xu et al. [261]	Alipour et al. [3] Jin et al. [99] Shin et al. [211] Li et al. [126] Li et al. [125]
UIA	Sen et al. [205] Kulkarni et al. [111]	Siblini et al. [212]	Hulburd [92], Otegi et al. [167] Aken et al. [2]	Le Berre et al. [19] Aken et al. [2]	-
SDDT	Zhang et al. [270] Kulkarni et al. [111]	Maitra et al. [141], Lockett et al. [136] Lee et al. [118], He et al. [82] Alloatti et al. [5] Sakata et al. [203]	Schwarzer et al. [204] Siblini et al. [212] Kumar et al. [112]	Li et al. [122] Habibi et al. [75]	Riley et al. [199]

Table 3.1: Work distribution over the dimensions identified by tasks and challenges of IQA systems.

Table 3.1 collects all the publications distributed among the previously described task-oriented classes. Although multiple works aim to realize an IQA system able to read and understand many textual documents to

answer users' questions, all analyzed research findings evenly occupy the highlighted categories. In addition, QA approaches vary depending on the challenges they target. We found different features in QA methods aiming for other goals, e.g., optimizing system performances (i.e., answers accuracy, response time, etc.) or the overall user experience. Therefore, QASs can be further classified based on the objectives they cover.

System Performances (SP). New approaches continually design novel QA systems with improved effectiveness compared to the ones composing the state-of-the-art. To this purpose, extensive offline experiments usually assess the SP on several public datasets.

User Experience, Trust, and Transparency (UTTP). QA systems aiming to solve this goal engage users more compliantly. Three main aspects are usually evaluated regarding users relationship with the system, that is *user experience*, *trustworthiness*, and *transparency*. Whether the first one deals with *satisfaction* and *usability*, the *trustworthiness* is more oriented to measure user expectations about results returned by the system, meaning how users trust system functionalities [144]. The *transparency* instead, evaluates the interpretability and comprehensibility of system processes [157].

Usability and Interaction Analysis (UIA). This challenge does not need a QA systems implementation. Its scope is to analyze the usage of already existing QA systems or users' behaviors toward interacting with them. Thus, statistical analysis is always provided and discussed for this goal [19].

Specific Domain-Dependent Task (SDDT). Most papers here design a QA environment to solve a specific issue. The proposed system usually drives users to execute domain-dependent processes or tasks more efficiently. Consequently, these solutions are designed and tested to fit the problem to solve instead of being generalized. This kind of publication often needs more comparisons with other state-of-the-art models [134].

Table 3.1 also depicts the distribution of publications among the challenges above, where a lack of balanced QA community interest emerges. Research efforts are almost totally focused on improving QA systems performances, while a very few works emphasize statistical analysis about usability and interaction. Implementation modalities and algorithms still fail to reach strong performances about not specialized QA systems. Thus, the IQA field still needs to be solved. To underline this evidence, we see a lower interest in User Experience, Trust, and Transparency challenges.

3.3 IQA Systems: Architecture and Techniques

Figure 3.2 outlines a high-level architecture with the components that an IQAS should implement. Four major components are proposed: the *interaction engine*, the *state tracker*, the *QA module*, and the *knowledge source*.

The **Interaction Engine** (IE) is the module that manages the interaction with the user, receiving her requests and providing the response to her. This module enables user and system interaction by leveraging different interfaces according to the system's capabilities. For instance, in case the user-system interaction is based on click (e.g. automaTA by Lee et al. [118]), the IE enables the system to capture the user request and prepare it in a suitable form for the *state tracker* (cf. Section 3.3.2). Similarly, when the interaction relies on natural language messages, then the IE supports the reception of textual messages [253], as well as images [199], or even videos [99] for visual user-systems interactions. The IE manages different *interaction modalities* according to the system implementation.

The complexity of the IE depends on several factors: (i) the *interactivity* level, (ii) the *operation modality*. The *interactivity* level refers to the distinction between standard and interactive QASs. The IE enables users to engage with the system in both *single-step* (e.g., QAS that returns the most relevant answer as a direct result to the user question [203]) or *multi-turn* interaction (e.g., systems allow further steps to refine or expand the given system response [221, 211]). The higher the interactivity level is, the more complex the IE-module implementation will be. The *operation modality* is de-

fined as the capability of the IE to deals with one or more *interaction modes*, making a IQAS *uni-modal* [185] and/or *multi-modal* [67]. Finally, the *initiative* defines whether the user [114], the system [261] or both [250] can trigger a disambiguation/exploration step, i.e. with disambiguation/follow-up questions.

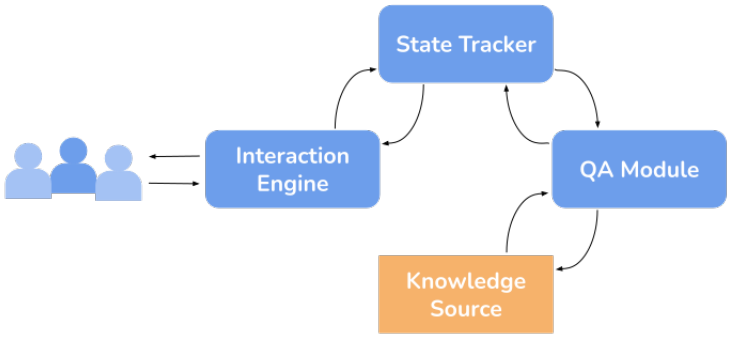


Figure 3.2: General Architecture of Interactive Question Answering Systems

The **State Tracker** (ST) manages all the information the *interaction engine* and *QA module* need in order to handle the QA request. In greater detail, it aims to fill up the *QA state* (cf. Definition 3.2.5) and to collect all the information exchanged between the user and system up to that time. In addition, referring to the Definitions 3.2.2, 3.2.3 and 3.2.7, an IQA task can also be seen as a *chain* of *interactive sessions* (cf. Definition 3.2.4) grouped by the user's information need. Accordingly, the ST keeps track of the exact point of the *chain* in which the system is at any given time: this is the *QA state* that represents a snapshot of the system at time t.

The ST operation mainly depends on two elements: (i) the *context* and (ii) the *tracking methods*. For CoQA systems, the same action could have different reactions and meanings given the *dialogue context*. In contrast, for *stateless* IQASs, the ST has no information about past interactions, dealing with *session context* as exploitable information (e.g., images or textual snippets) that completes the user input to the system.

With regard to *tracking methods*, in the literature they are classified as

implicit or *explicit*. An *explicit tracking method* keeps information about the previous *QA states* in a structured form. For example, natural language tokens or knowledge graph entities can be collected [256]. Conversely, *implicit tracking methods* store previous *QA states* in a latent shape like embeddings or trained parameters of a model. That is the case of systems that rely on deep learning models, which learn patterns on real dialogue sets and perform conversations without needing explicit state tracking techniques [188]. Here, the model learns the state as embedded representations.

The **Question Answering Module** (QAM) aims to provide the most suitable answer to the user question and represents the core component of every QA system. It retrieves or generates the response to the user by exploiting the context, if any, managed by the ST. The answer depends on the *data representation* adopted by the QAS (e.g., text, images, etc.) and the interaction mode.

In order to deal with them, the QA models can implement *data-driven* and *instruction-based* approaches. The former belongs to the ML field and encodes all the collected documents through *numerical representations* such as TF-IDF, one-hot encoding, Word2Vec, etc. [126]. The latter operates with both *categorical information*, such as pure text or KG entities [159], and *numerical representations* outlined before [130].

Finally, the **Knowledge Source** (KS) stores all the knowledge exploited by QAS to answer users' questions. The KS data can be *structured* (e.g KG, relational database, etc.) or *raw* (i.e. images, videos, texts, etc.). The data type can depend on the task the QAS aims to solve. For instance, a *factoid* QASs can rely on both *structured* information as a KG, a set of triples in the form of subject-predicate-object [209], and *raw format* data as a collection of textual documents [264].

In the following, we aim to give an in-depth snapshot of the state-of-the-art IQA systems for the components outlined in Figure 3.2. On this line, Table **??** groups some relevant examples provided by the literature according to the IQAS modules, their aspects, and technical specifications for their implementation.

Module	Aspect	Approach and Example work
Interaction Engine	Interactivity	single-step [19, 73], multi-round [103, 275].
	Operation Modality	
	• Single	text [127], audio [136].
	• Multiple	images & text [3, 210], audio & text [115], video & text [265].
	Initiative	system-driven [86, 272], user-driven [54, 187], mixed [46].
State Tracker	Tracking Methods	implicit [220], explicit [36, 134].
	Context	session [154], dialogue [188].
Question Answering	Image Representation	ResNet [89], Faster-RCNN [208], VGG Net [184], CNN [73].
	Text Representation	inverted index [43], pure text [65], term frequency [125], Word2Vec [89], Glove [208], LSTM [73], WordPiece [203].
	Model	
	• Instruction-based	keyword matching [43, 249], rules execution [36, 275].
	• Data driven	supervised [254, 123], reinforced [268, 47].
Knowledge Source	Structured	knowledge graph [218], relational database [159].
	Unstructured	texts [2], images [126], video [99].

Table 3.2: A summary of components in the IQAs representation pipeline, as sketched in Fig.3.2

3.3.1 Interaction Engine

The *interaction engine* manages the user-system interactions flow. Its main objective is to allow users to communicate with the system naturally. As a result, the IE uses interfaces tailored to the system's characteristics while properly formatting the information intended for the user. For example, CoQA systems require an IE that supports dialogues via chat-box interfaces. Rather than that, IQA systems that solve the VQA task require this component to attach photos as an additional feature. The IE can incorporate numerous system design choices, such as allowing users to ask follow-up questions (i.e., exploratory questions) or choosing a query from a predefined set. In a nutshell, the IE is characterized by three primary characteristics that include: *interactivity*, *operation modality*, and the *authorized initiative*.

The **interactivity** of a QA system determines how users engage in the

process of seeking answers. This aspect denotes the distinction between conventional QA systems and IQA systems. Traditionally, QA systems deliver immediate responses to user queries, excluding her from further discussions. In comparison, IQA systems accumulate several interactive sessions allowing users and the system to refine/enrich the initial result. In a nutshell, interactivity defines the QAS required to support interactive user sessions.

In the literature, systems that do not use interactive sessions to support achieved replies are referred to as *single-step* interactive systems. Such systems receive inquiries from the user and eventually supporting data (i.e., images, document extracts, audio tracks, etc.) to produce a unique and appropriate response. Thus, the system-user interaction concludes when the user receives the system response.

For example, Aken et al. [2] analyze how BERT-based QA systems answer user questions. Their system addresses the MRC QA task focusing on understanding BERT operations in retrieving the correct answer to a given query. Therefore, authors omit the system to enable *interactive sessions* for the computed solution, granting only *single-step* interactions with the user.

Conversely, CROWN, a CoQA system implemented by Kaiser et al. [103], exploits a supervised approach to allow context propagation through multi-round interactions. As a *stateful* IQAS, CROWN can understand the context left implicit by users during their conversations.

It is worth noticing that the IE *interactivity* has no direct relations with the *statefulness* property, which belongs to the *State Tracker*. Thus, the *interactivity* does not distinguish *stateless* IQA systems from *stateful* ones. As a further proof, the example in Figure 3.1 shows an IQA system that support multi-round interactions in a *stateless* configuration.

The QA system's **operation modality** specifies to the IE to deal with which kind of interaction mode. QA systems that manage more than one physical channel as a means of interaction (i.e., text and visual or speech and clicks) exploit a *multi-modal* IE. Instead, systems that allow unique *interaction modalities* in communicating with users (e.g. only text [230] or speech [115]) have their IE being *uni-modal*.

Qi et al. [185] implement a QA system based on *siamese networks* to learn user preferences and answer her subjective questions. To this goal, the QA system IE implements an *uni-modal operation modality*, which requires only textual questions.

Gordon et al. [73] address the VQA task through an autonomous agent that dynamically interacts with a visual environment to reach the answer. They design an IQA system that requires both a textual question and a visual context as input. Thus, the agent may seek other graphical views for disambiguating the user question or compute the answer employing several modules, i.e., a Scanner to capture images, a Navigator to change the system view, and a Manipulator.

However, the IE **initiative** designates the actor mainly involved in leading *interactive sessions*, which is three-folded. We refer to *user-driven initiative* when users can freely choose their questions as a follow-up to the system response (i.e., *exploratory question*). In contrast, the *system-driven initiative* specifies the *interactive session* being a finite set of data on which users may interact (e.g., picking a follow-up question from a collection computed by the system or answering *disambiguating questions*). The *mixed initiative* allows both the previous two cases.

Referring to Section 3.2, systems enabling the *disambiguation* have at most a *system-driven initiative*, while exploratory interactions can also include a *user-driven initiative*.

For example, the work of Su et al. [221] implements an IQAS which allows users to ask about the composition of API calls. In addition, it supports the user to refine the system answer by adding/removing parameters to the returned API call and exploring new related ones. The system provides these parameters as the *interactive session* related to the reached answer. Thus, the proposed system *initiative* is *system-driven*.

Finally, Google Assistant and the system implemented by Qu et al. [187] are CoQA systems that allow multiple user-defined interactions, which make *user-driven* their *initiative*. In both cases, the user is free to ask any exploratory question, moving the system to seek manifold information on a topic.

3.3.2 State Tracker

The *state tracker* is the module that deals with *QA states* as described in Definition 3.2.5. It supports communications between the IE and QA modules to solve the QA task as a central component. Thus, the ST takes data from both of them to update the *QA state* at each interaction step. It enables the QA module to exploit information (i.e., context data) in addition to the *knowledge source* for computing answers to user questions. At the same time, it supports the IE to configure the authorized interactions based on data currently stored in the *QA state*.

In literature, the ST strictly depends on the *statefulness* property of a QA system. It assumes different behaviours based on both *QA state tracking methods* and the type of *context* (cf. Definition 3.2.5). In particular, the *context* may differ in two types, i.e. *session context* and *dialogue context*.

The *session context* is a set of supporting data contained in the *QA state*, obtainable with just a single interaction with the user. The data hosted in this context depends on the QA system *operation modality* and the representation of that information. For instance, Shi et al. [210] design a Quaternion Block Network to solve the VQA task, which admits images in a numeric representation in its *session context*. Similarly, Das et al. [47] expect a set of textual paragraphs in a numerical form to support the user questions.

The *dialogue context* is instead typical of CoQA systems whose data grows after each interaction with the user. All the QA states feed the context of the next one, forming a *conversation* with the user. For example, IHAF is a CoQAS implemented by Perera et al. [171] that hosts textual data in form of a *conceptual graph* within the context of each *QA state*. During the conversation, it receives from users further information that expands the *contextual graph* at each interaction step, allowing the system to return more accurate answers the longer the conversation is.

To keep in memory all the *QA states*, the ST relies on **tracking methods**. In literature, we distinguish *implicit* from *explicit tracking methods* regardless the type of context hosted by *QA states* (i.e. session or dialogue context).

On the one hand, the *implicit tracking methods* deals with the storage

of *QA states* in numeric forms (e.g., latent representation). For instance, Su et al. [220] model a CoQA system as an adaptive framework based on the sequence-to-sequence model. It manages numerical representations of all the memorized *QA states* in a *conversation history*, that combined with current questions (also numerically represented), retrieves more straight answers.

Conversely, Chiang et al. [34] implement three CoQA systems with different *implicit tracking methods* to test their ability to comprehend textual contents regardless of their performances on the MRC QA task. They show how ML systems like FlowQA [91], BERT [52], and the proposed SDNet rely more on previous *QA state* answers instead of any contents host in the KS. In a nutshell, it computes answers by exclusion from the previous ones instead of being logically inferred from both the implicit tracked *dialogue context* and the KS.

On the other hand, the *explicit tracking methods* grant direct access to explicit data stored by the system. This information type is understandable by humans. Here, *QA states* have not a latent form, thus preserving the structure outlined in Definition 3.2.5. The system designed by Wong et al. [249] represents a first attempt to realize a CoQA system with an *explicit tracking method* for storing *QA states*. This solution expands the *dialogue context* with all the keywords that arose during the conversation with the user. Therefore, the *QA Model* infers appropriate answers relying on its KS and dialogue context, which hosts some weighted elements computed according to a decay function over conversation steps.

3.3.3 QA module

The core component of a QA system is the *QA Module*, whose goal is to find appropriate answers to user questions given a *knowledge source* and possible *contexts*. Thus, a *QA module* must understand user queries and infer the requested information from extensive data collection.

The *QAM* differs on the type of data the QA system has to deal with and the modeling strategies that fit the QA task resolution. Therefore, it is

described by two characteristics, i.e. the *data representation* and the *modeling approach*.

The **data representation** depends on the supported *interaction modality* and its *modeling approach*, which in turn counts on the information types hold by the *Knowledge Source* (cf. Section 3.3.4). In fact, we distinguish *categorical* data (e.g., a sequence of textual strings) from *numerical* one (e.g., vectors as well as matrices). It is worth noting that the QAM *data representation* is the data model that best fits the requirements of a QAS, which may differ from the ones of the *knowledge source*. In a nutshell, the QAM supports a *numerical data representation* when its model requires data being expressed with numeric elements to compute an answer (e.g., dealing with images/audio or deep learning architectures). For example, the work of Hu et al. [89] enriches the semantic information of possible answers by adding images to their knowledge source. Thus, the QAM requires a *numeric data representation* to evaluate semantic similarities between queries and possible answers.

Mandya et al. [143] focus on the Co-reference resolution task in a conversational setting through ML models. In particular, they highlight a set of co-reference question-answer chains in the *dialogue context* searching for terms that may refer to the same entities of the user question. Their implemented model automatically achieves the goal by employing attention mechanisms, which requires data in a *numerical* form.

On the other hand, Fukumoto et al. [65] leverage the conversational issues through matching keyword methods and Named Entity Recognizer. Thus, their QAM relies on data expressed with their *categorical* values.

The **Modeling Approach** identifies the strategies adopted by the QA system to make it practical and effective to reach an answer given a question. In other words, it refers to the framework that existing algorithms or new ones exploit to solve a QA task. The choice of a model relies on three factors, which are the QA task to be solved (cf. Section 3.2.3), the supported functionalities (e.g., targeted challenges or the allowed interactivity), and the available data (i.e., the *knowledge source*). Nevertheless, each implementation can be classified as an *instructions-based* or *data-driven* model. The

former takes advantage of well-designed instruction sets for its goal, while the latter tries to take out and learn patterns from a huge set of data (i.e., query-response pairs) to reply to the user questions.

Hence, the *instructions-based* methods rely on a set of rules designed a priori to accomplish its task. The resulting QAM is unaware of data held by the *knowledge source*. Thus, its implementation covers a finite number of scenarios that may emerge while answering questions. The state-of-the-art QASs can be further divided into three categories: (i) *keyword matching* models, (ii) *pipeline execution* algorithms an (iii) *translation* models.

The *keyword matching* strategy aims to find the answer to a question based on the number of matches between their key terms. For instance, the previous work of Schwarzer et al. [204] implements several scoring functions that retrieve the answer among the most relevant document based on how many question keywords the system found.

The *pipeline execution* models foresee a sequence of functions performed in cascade to solve the QA task. Christmann et al. [36] take advantage of this modeling strategy to implement Convex, a *factoid* CoQA system based on the Wikidata[3]. The authors design a set of processing steps to be executed in the pipeline for solving the KB QA task. It foresees a (i) named entity recognition and disambiguation (NERD) system; a (ii) context subgraph built upon Wikidata entities recovered by (i); (iii) key entities retrieved based on three relevance evaluations; an aggregation through (iv) the Fagin's Threshold Algorithm [61], and the answer obtained from (v) a scored list of entities.

Furthermore, *translation* approaches to perform a translation of questions from a natural language to a formal one, which depends on the QA system *knowledge source*. It allows users direct access to complex structured data collections (e.g., knowledge graphs or relational databases) without being an expert in the related query languages (i.e., SPARQL and SQL). Naeem et al. [159] implement a QA system that translates the natural language user questions into OLAP queries, while FREyA, by Damljanovic et

[3]https://www.wikidata.org/

al. [46], is one of the first examples of KB QA systems that translates the user questions into SPARQL queries. It identifies a set of ontology concepts from the natural language question by combining syntactic parsing and ontology reasoning techniques or using string similarities evaluation, synonyms detection, and user engagement. The SPARQL query is built based on the retrieved ontology concepts.

Differently, *data-driven* strategies refer to models trained on often huge data collections to learn patterns for answering user questions. These models build their knowledge by looking at the examples provided during a training phase. Then, they take advantage of their training to solve a specific task (i.e., answering questions). The QAM *data-driven* strategies found in the literature are classified in *supervised* and *reinforced*.

The *supervised data-driven* model relies on a labeled dataset to learn the answers linked to a given question example, divided into *detecting* and *generative* implementation, which depends on the method computing the answer. With the **detecting** approach, the QAM learns how to detect the desired response from an information set. Thus, answers can be found as a limited sequence of words within a document or picked from a set of answers held by the knowledge source. For example, the work of Alloatti et al. [5] proposes a CB QA system on the e-invoicing domain and its regulation. The authors opt for a BERT-based model with an added layer to classify user questions into specific groups, which are, in turn, labeled with the related answer. The training procedure matches BERT's fine-tuning in learning a one-hot encoded vector whose indexes refer to the different groups of answers.

In contrast, *generative* approaches build the answer by composing appropriate words to the given question. The output comes from scratch instead of being selected among the existing ones.

Li et al. [123] train a CoQA system to answer the user's question about information previously given to the system. It expects a list of sentences defining a queryable "story" as input. Then the user asks for data about that story in a conversational setting. The system also learns to pose disambiguation questions to the user when its current knowledge lacks crucial

information for computing a reply. The *generative data-driven* model encodes both sentences and questions via Gated Recurrent Unit (GRU). Then, they decoded such representations into answer/disambiguation questions.

Finally, the *reinforced data-driven* models adopt the paradigm of *reinforced learning* to comprehend the QA task. Hence, the QAM performs some *actions* based on *observations* and *states*, which may results in *transitions* to other *states* eventually *rewarded*. The training procedure states the system attempting to solve the QA problem through an intrinsic logic. When the QA system answers correctly, it receives a reward that usually minimizes its training cost function. Otherwise, no incentives, or penalties, are given to the system.

Gordon et al. [73] propose a visual QA system modeled as an agent able to explore the environment of a given picture through several actions (i.e., navigating, manipulating, scanning, detecting, and answering). Here, the authors enable the hierarchical reinforced learning paradigm to train a high-level controller in selecting and invoking the right sub-task (i.e., the previous actions) to reach the correct answer efficiently. The reward is received when the system planning produces a positive outcome (i.e., the correct answer).

3.3.4 Knowledge source

The **knowledge source** (KS) collects the information the system needs to compute answers to user questions. Data about facts, domain-specific instructions, and community opinions present different forms and modalities. This component represents the bases on which the QAS knowledge is built, defining what it is aware of and does not know. The hold information is made available both at the running time and at an eventual training time for the QAS. This distinction determines the type of the adopted QAM modeling approach (cf. Section 3.3.3), which also depends on the kind of data contained in the KS.

Although the KS module usually mirrors the main features of datasets exploited by the system to accomplish its task (cf. Section 3.4), in this sec-

tion, we provide a view oriented to the QA processes instead of its data. We focus on the organization of information collections at a high level of abstraction to label the state-of-the-art QASs approaching methods with data. The QAS *knowledge source* can be classified into *structured*, *unstructured*, and *mixed*.

The **structured** KS manages the information of the system knowledge through well-designed structures. In other words, it accepts only data that presents a standard structured organization universally recognized (e.g., knowledge graphs, relational databases). This is the case of Zheng et al. [275] and Zhang et al. [272], where the implemented systems are designed to rely on knowledge graphs (i.e. Dbpedia [149] and Freebase [23]) to retrieve the answer. Instead, the collection of question-answer string pairs (e.g., QALD-6 and WebQuestion datasets) evaluates only the system's performances. Li et al. [122] instead foresee a relational database as their CoQA system *structured* KS.

In contrast, **unstructured** KSs allow data sources to lack an intrinsic structure. They cover most QA systems populating today's literature, relying on paragraph lists or image collections to reply to user queries. Wu et al. [253] implement a cQA that depends on question-answer pairs (i.e., string sentences) retrieved from forum websites. Instead, Gao et al. [67] use a list of question-images pairs labeled with answers as a KS for their VQA system.

Finally, the **mixed** KS enables the system to rely on both structured and unstructured data sources. An example is given by Zhang et al. [273], where the implemented QA systems rely on multi-modal KGs (with images) and collections of qualified doctor advice to consultants in the form of question-answer strings to build their knowledge. Shen et al. [209] also exploit structured and unstructured information to train their CoQAS. In this case, authors use the Wikidata KG for the semantic knowledge and the Complex Sequential Question Answering (CSQA) [202] dataset to deal with dialog.

3.4 Evaluation and Dataset

While all IQA systems in the literature are characterized using the unified architecture seen in Figure 3.2, the datasets used to train and evaluate their models do not share a standard set of properties. Depending on the QAS purpose, the modality type, and the interactivity setting (i.e., QA, IQA^e, IQA^d or CoQA), the nature and characteristics of a dataset might be vastly different. Consequently, evaluation methods also end up being rather distinct.

While all IQA systems in the literature are characterized using the unified architecture seen in Figure 3.2, the datasets used to train and evaluate their models do not share a standard set of properties. Depending on the QAS purpose, the modality type, and the interactivity setting (i.e., QA, IQA^e, IQA^d or CoQA), the nature and characteristics of a dataset might be vastly different. Consequently, evaluation methods also end up being rather distinct.

The evaluation phase of IQA systems determines the system's efficacy regarding the intended task and challenge. Permitting comparisons between current solutions is essential, therefore identifying the most effective for a given objective. The evaluation step is critical to advancing the state-of-the-art.

There are two types of evaluation protocols: *offline* and *online* which are used for different types of evaluations. Similarly to ML/RS models, *offline* evaluations for IQA systems use pre-compiled offline datasets. Conversely, *online* tests require the system to work with real users to evaluate its functioning and returned results.

However, concurrently offline and online configurations realize a further detailed system evaluation. For instance, Zhang et al. [272] and Shin et al. [211] implement this mixed strategy to evaluate their systems in a broader range of features. The former engaged 300 college students to analyze the system performance changes in increasing the number of hints given by human participants. The latter asked 3000 workers to evaluate response features like diversity, attractiveness, and expressiveness.

Task	Modeling Approach	Dataset
cQA	Data Driven Instruction Based	Yahoo!, StackExchange
CB QA	Data Driven Instruction Based	Domain Dependant Datasets
MRC QA	Data Driven Instruction Based	CoQA, SQUAD, TriviaQA, SearchQA CLEF, QuAC, SQUAD
KB QA	Data Driven Instruction Based	SQUAD, CSQA WebQuesitons, QALD-N, SimpleQuestions
Visual QA	Data Driven	VQA, VQA2.0, TDIUC, COCO-QA

Table 3.3: Main relevant datasets exploited in IQA systems literature grouped by tasks and QAM types.

In detail, the evaluation focuses on the following:

Single Answer. Systems are assessed based on their ability to respond to user queries, with the returned responses serving as the primary evaluation metric. Depending on the problems, tasks, and methodologies of the QAS, solutions can be assessed at several levels, which in turn establish assessment objectives such as *correctness*, *reliability*, and *sensitivity*, as well the *naturalness* and the *expressiveness*.

Ranked List This group refers to the system's capacity to retrieve appropriate resources for a given query. Although the QA task allows a single result to return to the user, some state-of-the-art works also permit other replies produced by the system after the initial response. This list reveals the *reasoning capabilities* of the QAS and which features/information are deemed necessary for locating appropriate responses. The examined test objectives are comparable to those of the first group, although being able to be evaluated on a deeper level (e.g., answers positions in the returned ranked list).

Interaction Rather than the solution itself, the attention is on the interaction enabled by the system to arrive at it. The interaction influences the quality of the system's responses. Consequently, interactions are assessed based on their *cost* and their required *user efforts*, as well as

their *effectiveness* and *efficacy*. The majority of studies focusing on this element employ online evaluation techniques. Nonetheless, offline metrics are also utilized for stateful IQA systems, which may further evaluate the coherence, context, and naturalness.

The table **??** shows the most frequently used datasets for building, testing, and evaluating an IQAS according to the literature.

3.5 Conclusion

We have reviewed a substantial collection of interactive question-answering (IQA) systems-related literature published during the past decade. We discovered the literature to be diverse, beginning with adopted methodologies for addressing multiple QA tasks and concluding with a vast array of resources (i.e., knowledge sources and datasets) utilized to create and evaluate question-answering (QA) systems. Although several types of QA solutions define the state-of-the-art, we determined the characteristics shared by the suggested systems that constitute a shared framework. To the best of our knowledge, we are the first to present a unified and comprehensive design that emphasizes the fundamental components and functions of IQA systems. For each element, we have comprehensively categorized the literature from the methodological and application perspectives, categorizing the works by tasks, difficulties, and interaction modes. In addition, we give explicit definitions of the implementation goals and features of IQA systems. To achieve this goal, we have categorized QA systems based on their behaviors and enabled interaction so that we captured the whole IQA systems landscape. Then, we detailed trends regarding specific tasks and problems, demonstrating the community's keen interest in enhancing the system's performance and openness. Lastly, we have included a classification of evaluation approaches often used in the literature and a list of primary datasets used in the evaluation process.

4

Interaction with Aspect-Based Sentiment Analysis

Nowadays, several social networks and Web platforms permit users to share their opinions and tastes on items of different types. As a result, there is a growth of data relating to the subjective sphere of each individual. This information is handy for several purposes, such as providing personalized recommendation services or understanding opinions conveyed through text. Sentiment Analysis provides helpful methods to analyze these textual opinions (e.g., reviews) from a global point of view. In case we want a more detailed representation of the opinion represented in a text, Aspect-based Sentiment Analysis identifies a valuable option thanks to its fine-grained level of text analysis.

In this Chapter, we discuss the design of a processing pipeline to extract domain-related aspects from the text by employing an unsupervised approach. We formally define Aspect Terms and Aspect Categories and Aspect-based Sentiment Embedding, an approach to represent documents by computing aggregated sentiment scores for each aspect. We perform experimental evaluations on the Spotify dataset to prove the utility of our technique in predicting elements strictly related to emotions and feelings. Our results show improvements in the regression task for sentiment-related

features compared to the classical semantic-based representations.

4.1 Introduction

The increasing amount of user-generated content on blogs, social networks, and e-commerce websites has pushed many companies to explore data mining technologies to use this subjective information source. One of the most prolific research areas in Natural Language Processing is Sentiment Analysis, which aims to identify and extract user opinions from the text. Sentiment Analysis has the goal of identifying sentiments expressed in texts and whether the expressions indicate positive (favorable) or negative (unfavorable) opinions toward the subject [161]. For instance, a sentence like *"The latest Apple Macbook has finally arrived and is amazing!"* tweeted by a user can give the Apple Company a valuable opinion about its product just released. Another example might be the detection of the emotional state of a customer during a dialog with a chatbot. Automatic customer service may improve its efficacy by moving the dialogue to a human operator in case the conversation becomes frustrating for the user. Sentiment Analysis techniques cover all those situations, finding and measuring the subjectivity behind words that express a Sentiment.

Sentiment Analysis can be performed at three levels: document, sentence, and entity. While at both document and sentence levels, sentiment analysis aims to find the overall opinion expressed in a text, entity-based sentiment analysis aims to discover the opinion of a user with respect to a specific entity extracted from the raw text. This task is typically known as Aspect-Based Sentiment Analysis (ABSA). More specifically, ABSA mines opinions from the text about specific entities and their aspects [181]. It involves two main sub-tasks: the Aspect Term Extraction (ATE), which extracts all the words that refer to an aspect, and the Opinion Term Extraction (OTE), which aims to find all the expressions that convey the user opinion concerning the extracted aspect terms (i.e., assign a sentiment score to a specific aspect term).

The task of ABSA is not trivial because of different reasons. There is no

objective and formal definitions of *Aspect*, and, in general, possible aspects differ from one domain to another. Moreover, users could refer to the same concept using different words or expressions. For example, in the Semeval 2014 laptop dataset [180] there are a lot of training samples that express opinions about the aspect *"Technical Support"*, but reviewers refer to it with expressions like *"technical service"*, *"tech support"* or *"staff"*. Additionally, there are no consistent and labeled datasets for the ABSA task, pushing the majority of work in this field to adopt unsupervised approaches.

Nevertheless, a vast amount of data about sentiment conveyed by text is available, given the high interest of the community in automatically understanding subjective opinions and emotions from written documents. Among several domains, datasets belonging to the music field undoubtedly offer the wealthiest collection of texts. On this line, several works started to analyze and exploit that data for several purposes, like recommending music based on users' emotional states [6] as well as analyzing how songs characteristics match user personalities [152].

In this chapter, we aim to answer and give empirical evidence to the following research questions:

- *Is it possible to group Aspect Terms extracted through unsupervised approaches into coherent Domain Aspect Categories relevant in an ABSA scenario?*

- *Can a Sentiment-based text representation improve the performances of a Sentiment-related task?*

To reach these goals, the unsupervised paradigm chosen for the aspect detection task exploits dependency trees and syntactic relations between words to extract not only *Aspect Terms* but also *Opinion Words*, allowing us to assign a sentiment score to them. Therefore, we adopt a Word2Vec representation to group all the similar *Aspect Terms* and define the *Aspect Categories* they belong.

As the main contribution, we introduce an unsupervised method to extract aspects from text which are relevant for Sentiment Analysis. Com-

pared to other solutions, which detect the *Aspect Terms* but compute a sentiment score for the overall document, our solution groups all the found *Aspect Terms* into *Aspect Categories*, each of them characterized by a specific sentiment value. To highlight these distinctions, we also provide formal definitions about *Aspect Term* and *Aspect Category*. Moreover, we introduce the concept of *Aspect-Based Sentiment Embedding*, a representation strategy of textual documents based on domain aspects and their related sentiment scores. We demonstrate the utility of this embedding technique by designing an experiment aiming to prove that our approach grants improved performances in predicting textual features related to human feelings against the well-known semantic approaches.

Our intuition is that a sentiment-based embedding could be more representative than a semantic one for text rich in sentiment. To the best of our knowledge, this is the first attempt that introduces the concept of numerical representation based on opinion and exploits it to improve performances of sentiment-related tasks.

Since song lyrics are rich in human emotions [35], we choose the music domain to evaluate our sentiment-based representation on a prediction task. In particular, Spotify, one of the most renowned music streaming services, released a dataset composed of features that best fit our needs paired with song lyrics. To prove our hypothesis, we train several models for predicting different elements on a given lyrics representation based on sentiment (Aspect Based Sentiment Embedding) or semantic vectors (i.e., Doc2Vec). This comparison highlights the latter relies more on semantic features like word co-occurrence or context that embed the semantic meaning of each word, and the former encloses information strictly related to subjective sentiment and opinions. In other words, the semantic embedding identifies what the user means, while the sentiment one points out what the user feels. Therefore, our proposed sentiment-based embedding strategy is more suitable for assessing features strictly related to human emotions. Finally, we assessed the statistical significance of our results with a T-Student test.

4.2 Unsupervised Aspects Extraction

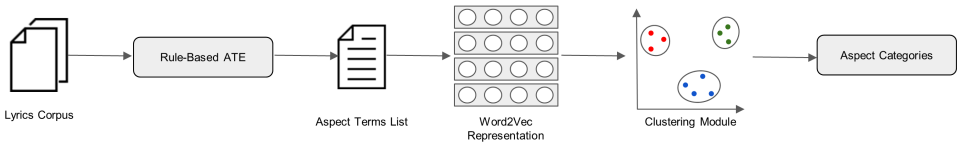


Figure 4.1: Aspect Term Extraction and Clustering

One of the main issues we address is that aspect terms extracted through state-of-the-art approaches are not grouped into categories since they are not inferred. Another problem is that these works do not leverage all the fine-grained sentiment information enclosed in each Aspect. Here, we address the problem of unsupervised ABSA for extracting relevant aspects and related sentiment to understand if it is possible to infer better features related to human feelings with a representation based on sentiment data instead of semantic one. For this purpose, we implement an unsupervised ABSA model that has been tested on the Spotify dataset, creating a representation of each song based on the extracted aspects and the related sentiment. We use these song encodings to predict scores like danceability, valence, mode, and others, which we will discuss in the following sections.

To the best of our knowledge, there are no works in literature that exploit aspects and their related sentiment for a regression task like the one described before.

Finding relevant aspects in a specific domain is not a trivial task since they are not known a priori and because of their intrinsic subjectivity. Several NLP techniques can infer these concepts that typically describe features of items like instruments or cars if they still need to be provided.

To reach this objective, one possibility is to collect all the nouns that are directly related to words that carry a sentiment with them (e.g. *unconditional love*, *great party*, *beautiful sun*, etc.). The main intuition behind this assumption is that items aspects are generally evaluated by the reviewers who expose opinions on them [69, 81].

Without loss of generality, it is possible to widen this assumption to any

written text belonging to any domain. For example, in the music domain, topics treated by the songwriter identify sentiment-related aspects. Therefore, depending on the matters and their sentiment polarity, we can infer some peculiar features more related to subjective emotions instead of the semantic meaning.

Given a set of documents, we define the pipeline based on an unsupervised approach depicted in Figure 4.1, able to identify all those aspects that characterize a specific domain. This choice has been made given the absence of a complete and consistent dataset designed for this task and to keep the aspect extraction algorithm as less domain-dependent as possible. To better explain how the proposed method reaches the final goal of detecting the domain aspects, we found it helpful to introduce two formal definitions of the *Aspect Category* and the *Aspect Term*.

Definition 4.2.1 (Aspect Category)**.** Given a domain D, an Aspect Category A, or more briefly Aspect, is either a topic or concept which meticulously describes a specific domain characteristic s.t. a domain is portrayed by a set of N Aspects $D = \{A_0, A_1, ...A_N\}$. □

As an example, we may consider the domain of movies. Some Aspect Categories which best portray film industry products are undoubtedly the ones evaluated by the Academy of Motion Picture Arts and Sciences assigning the Academia Award, like *Picture*, *Visual Effect*, *Director*, etc. Then, for the movie domain and the corresponding Aspects are $D = \{Picture, Director, Visual\_Effect\}$

Definition 4.2.2 (Aspect Term)**.** Let A be an Aspect for the domain D and w a generic natural language word. Then w is an Aspect Term for D if w is a token used to denote the concept or the topic of the Aspect A. We denote this relation with $w \triangleleft A$. □

Following the previous example, a movie can be entirely described by talking about its computer graphic, the plot, and the actors' performances, all words that belong to the Aspect Terms sets of the movie domain. For instance:

$$(light, color, saturation, landscape) \triangleleft Picture$$

$$(monster, fire, explosion, dragon) \triangleleft Visual\_Effect$$

$$(adaptation, framing, casting) \triangleleft Director$$

Therefore, given the sentence: *"Interstellar is a film with incredible landscapes, where realistic space and planets are the contexts for great actors and an impeccable soundtrack."*, we can infer that:

$$D = \{Picture, Visual\_Effect, Actor, Music\}$$

$$(landscape) \triangleleft Picture \wedge (soundtrack) \triangleleft Music$$

$$(space, planets) \triangleleft Visual\_Effect \wedge (actor) \triangleleft Actor$$

The main goal of our pipeline is the Aspect Term Extraction. Based on a sequence of processing steps, starting from textual data, it can identify *Aspect Categories* together with the corresponding *Aspect Terms*. Following the process depicted in Figure 4.1, we have a sequence of steps:

1. **Candidate Aspect Term Detection:** all the words that have a high probability of being an *Aspect Term* are here identified through the *Rule-Based Aspect Term Extractor*. It searches entities related to opinions exploiting some syntactic rules. In detail, nouns usually refer to entities, while adjectives, adverbs, and some verbs usually carry sentiment information that defines the opinion polarity [252]. Thus, only those nouns strictly linked to words having a non-neutral sentiment score are selected. For this purpose, we generate a Dependency Parse Tree for all the sentences in each collection document since dependency relations reveal connections between words and possible sentiment inflections.

   All nouns that have dependencies with adjectives, adverbs, adjectival modifier (amod), and adverbial modifier (advmod) identify the *Candidate Aspect Term*, including also nouns linked to verbs with a non-neutral sentiment score obtained from the Sentiment Lexicon SentiWords [68]. For instance, expressions like *"The colors make you love the film"* give a positive opinion on the aspect *Picture* with the composed

verb *makes - love* without exploiting adjectives or adverbs. It is worth noting that words enclosing opinions are considered a filter condition to select candidate terms of interest without storing any information about them. The frequency of each unique term on the entire collection of documents is kept.

2. **Aspect Term List Generation:** From the *Candidate Aspect Terms*, all the words that have the same *lemma* are collapsed via lemmatization. For each term that shares the same lemma, we sum its relative number of occurrences. Then we delete all terms whose counter is lower than a specific threshold (10 in our case), resulting in the final *Aspect Terms List*, as outlined in Figure 4.1. This choice avoids taking into account aspects that are not highly informative for inferring *Aspect Categories*, thus generating noise in the following steps.

3. **Word2Vec Representation**: aiming to define which are the main *Aspects* that qualify the domain of interest, we take advantage of semantic vector representations of terms like Word2Vec [151]. Due to its known capability of shaping categorical features like words into numerical representations with an intrinsic semantic spatial distribution, we found Word2Vec particularly fits our goal. We compute for each lemma that populates the *Aspect Terms List* the Word2Vec embedding.

4. **Aspect Categories Extraction**: all the Aspect Terms embeddings are grouped in clusters using the K-means algorithm [140]. It allows us to define those categories which best assemble the *Aspect Terms* in a semantically consistent way. The *Aspect Categories* are the *centroids* of the retrieved clusters, and each of them keeps track of a list of *Aspect Terms* to which they belong.

To clarify the process of Aspect Term Extraction and Clustering, shall we consider two example tracks from the music domain: *"My Father's Eyes"* by Eric Clapton and *"Civil War"* by Guns 'N Roses. From these two songs,

our approach can extract some aspects such as:

$$D = \{Family, Love, War, Humanity\}$$

More in-depth, to define the previous *Aspect Categories*, the Rule-Based ATE module receives as input a set of lyrics, whose excerpts are: *"When I look in my father's eyes, my father's eyes, then the light begins to shine, and I hear those ancient lullabies"* and *"I don't need your civil war, it feeds the rich while it buries the poor, your power hungry sellin' soldiers, in a human grocery store"*. Then, it extracts some candidate *Aspect Terms* like *"father"*, *"lullabies"*, *"war"*, *"power"* and *"soldiers"*. After removing not frequent terms like *"lullabies"* and *"power"*, we obtain that the remaining words belong to the Aspect Term List and their Word2Vec representations. These representations are then used by the Clustering module, together with all the other aspect terms extracted from the lyrics, to define the previous Aspect Categories, reaching:

$$(father) \triangleleft Family \wedge (war, soldier) \triangleleft War$$

It is worth noting that our *Aspect Extraction* approach is not domain-dependant. Without loss of generality, we can apply our method to several objectives in many domains, like reviewing items for an e-commerce platform or detecting sentiment features in a song collection.

4.3 Sentiment Representation

The main idea behind our work is to identify a numerical sentiment-based representation of documents, which could deal with features strictly related to emotions or opinions more accurately.

To define this new kind of embedding, we found it helpful to set the *Aspect Categories* as dimensions of the vector representation, in the sense that each domain aspect match with a specific vector position whose element or value enclose sentiment information about that Aspect. For instance,

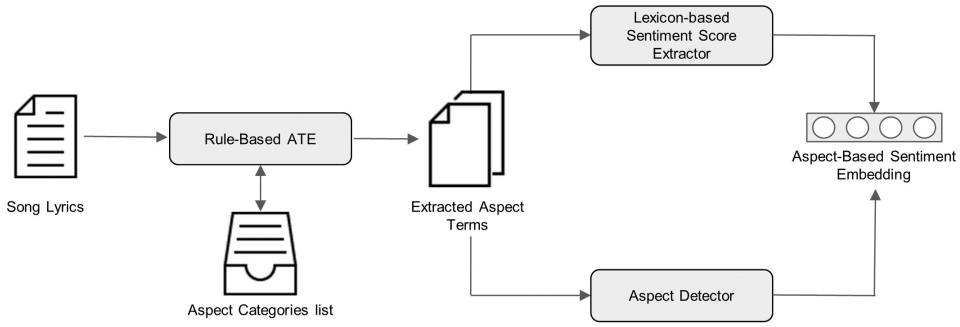


Figure 4.2: Aspect Based sentiment representation strategy

assuming that we found fifty *Aspect Categories* in the music domain:

$$D = \{People, Friendship, ..., Love, ..., Time\}$$

Sentiment embeddings of songs lyrics will have the first position referred to the aspect *People*, the second one related to *Friendship*, the forty-eighth to *Love* and the last to *Time*. Each of them could have a non-zero value depending on whether the songwriter expresses an opinion on the Aspect or not, as also highlighted in Figure 4.3. More formally:

Definition 4.3.1 (Aspect-Based Sentiment Embedding)**.** Let A be a set of Aspect Categories, N the cardinality of A and s a sentiment score s.t. $s \in [-1, 1]$. Given a document D, composed of different sentences, we define the Aspect-Based Sentiment Embedding $\mathbf{e}$ of D the vector:

$$\mathbf{e} = (s_0, s_1, ..., s_N)$$

where $s_i, i = 0, 1, ..., N$ is the sentiment score computed on the i-th Aspect Category.

Documents that treat the same aspects are similar in their representations. Moreover, depending on the opinions expressed for each Aspect, the computed sentiment value is an aggregation of sentiment scores obtained from each adjective, adverb, and verb linked to that Aspect.

More in-depth, we base our sentiment-based representation on the results produced by the *Aspect Term Extractor* presented in Section 4.2. Starting from the list of *Aspect Categories* that our model found over a domain, our approach takes as input a document written in Natural Language and searches for aspects and the related sentiment scores to shape the aspect-based sentiment embeddings. The designed pipeline is outlined in Figure 4.2.

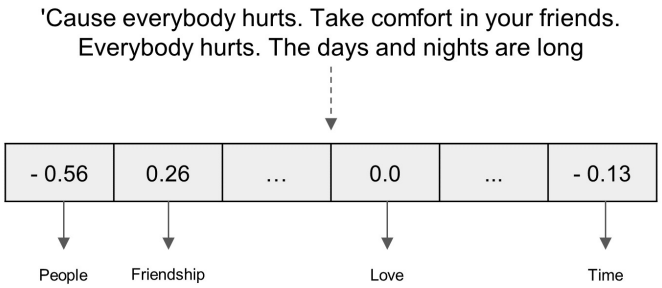


Figure 4.3: Aspect-Based Sentiment Embedding example

To evaluate our system, we choose to build aspect-based sentiment embeddings over documents belonging to the music domain because song lyrics carry a more emotional meaning instead a semantic one. Hence, they fit to test performances in predicting sentiment-based features. However, our approach is not dependent on the music domain, thus relying on a field-independent method that exploits an *Aspect Category* list to identify the numerical representations.

The proposed method receives in input a document, a song in this case, and searches for *Extracted Aspect Terms* by employing the *Rule-Based Aspect Term Extractor* already adopted in the *Aspect Term Extractor* outlined in Figure 4.1. Here, terms with sentiment inflections are collected together with adjectives, adverbs, and non-neutral verbs to specify their sentiment value. For each aspect term, we go back to its lemma and compute its Word2Vec representation.

As a consequence, the *Aspect Detector* module, shown in Figure 4.2, estimates the cosine similarity between all the aspect terms and the centroids of each aspect category stored in the *Aspect Category List*. This way, we can

assign each aspect word to a specific *Aspect Category* based on the highest similarity value. At the same time, we delete those terms whose similarity scores are lower than a certain threshold. As a result, we produce a list of *Aspect Terms* and related sentiment words assigned to specific *Aspect Categories*.

We can now build the *Aspect-Based Sentiment Embedding*, whose dimension is equal to the number of categories composing the *Aspect Categories* list. Each position of the embedding vector matches with a specific *Aspect Category*, while the value is computed based on the sentiment score related to that Aspect. To this scope, we implement a Lexicon-based Sentiment Score Extractor relying on the Sentiment Lexicon SentiWords, a high coverage resource containing roughly 155,000 English words associated with a sentiment score between -1 and 1 [68], to retrieve the ones related to words carrying the opinion of the writer. It follows that the overall sentiment value for an aspect is an aggregation of scores obtained for each aspect term belonging to that aspect category. In our case, the aggregation function we adopted is the simple summing function that sums all the sentiments of adjectives, adverbs, and non-neutral verbs related to each aspect term belonging to the same category.

The Aspect-Based Sentiment Embedding built with this approach follows an intuition similar to the one that characterizes the one-hot encoding representation. While in the one-hot encoding embedding, each word identifies a specific position in the vector, in our approach, the aspect categories define the space of the embedding vectors. In this way, documents that treat similar aspects with a similar sentiment share the same emotional meaning.

4.4 Experiment

This section describes in-vitro experiments we have run to understand if some music track features in the Spotify dataset related to the feelings conveyed through songs can be better inferred using a representation based on the sentiment instead of one based on semantics.

We have used a free dataset on Kaggle[1] that contains various types of information over more than 18,000 Spotify songs, including artist, album, audio features (e.g., loudness), lyrics, the language, genres, and sub-genres. The dataset contains many songs from a period from 1921 to 2020.

From the dataset, we have selected only songs with lyrics in English, discarding the instrumental ones because they are not exploitable by our model. We have split the dataset using the following proportions: 60% for the training set, 20% for the validation set, and 20% for the test set. The first step was to feed the training to the pipeline described in Section 4.2. In this way, we could extract all the aspect terms from lyrics and group them into aspect categories, identifying the main aspects of the music domain. We have adopted a Word2Vec representation of aspect terms exploiting the model trained on Google-News dataset[2] and then we have used the K-means algorithm to cluster the aspect terms into their categories, setting the number of clusters to 75 resulting from the Elbow method [226].

Once we defined the aspect categories, we proceeded with the online step to estimate the Sentiment-Based Aspect Embedding for each song in the dataset.

As stated in Section 4.3, song lyrics were the input to the Rule-Based ATE module for extracting candidate *Aspect Terms* and related opinion words from the text. For each candidate, we compute similarity scores with the centroids of all the *Aspect Categories*, keeping only those values greater than a certain threshold, empirically set to 0.4. The highest score identifies the cluster to which the candidate belongs. At the same time, we calculated the related sentiment using the SentiWords sentiment lexicon. Following this approach, we were able to represent a song as a vector of 75 elements in which the element at the position i represents the sentiment expressed in the text for the i-th *Aspect Category*, according to the Definition 4.3.1.

The main objective of our work is to answer the following research question: *"What is the best representation for each Spotify feature? The semantic or the sentiment-based one?"* To this purpose, we decide to use Doc2Vec

[1]https://www.kaggle.com/imuhammad/audio-features-and-lyrics-of-spotify-songs
[2]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/

[117] as our baseline for the semantic-based representation of song lyrics. To validate our hypothesis, we have trained regression models on both sentiment and semantic songs representations to predict the following six key features in the Spotify dataset (for a total of 12 models), whose values are in a range from 0 to 1:

- **Valence** describes the musical positiveness conveyed by a track.
- **Mode** indicates the modality (major or minor) of a track, the scale of its melodic content.
- **Speechiness** detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g., talk show, audiobook, poetry), the closer to 1.0 the attribute value.
- **Acousticness** is a confidence measure of whether the track is acoustic
- **Liveness** detects the presence of an audience in the recording.
- **Danceability** describes how suitable a track is for dancing based on a combination of musical elements

For completeness, we have also trained the regression models for Speechiness, Acousticness, Liveness, and Danceability, even if those variables are unrelated to the sentiment. Speechiness represents how much the track is wordy, while the others identify the technical audio characteristics. On the contrary, Valence and Mode are more linked to the sentiment conveyed by song texts. The Valence is the positiveness of a song, in the sense that tracks with a high valence score sound more positive (happy, cheerful, euphoric), while tracks with a low valence value sound more negative (sad, depressed, angry). The Mode is the likelihood of a song being in a major or minor scale. Even though this description seems unrelated to sentiment, different studies demonstrated that minor scales convey a negative mood while major ones communicate a positive emotion [109]. The model's architecture adopted for solving the regression task is a Fully Connected Neural Network with three hidden layers with ReLU activation and an output

layer with a linear activation function. We used the early stopping strategy to optimize the training and avoid overfitting by monitoring the Mean Absolute Error (MAE) and Mean Squared Error (MSE). We have also trained a Doc2Vec model on the song lyrics to obtain those semantic-based representations to use as input in our model.

Spotify Feature	Representation	MSE	MAE
Valence	doc2vec	0.0542	**0.1891**
	sentiment	**0.0530**	0.1920
Mode	doc2vec	0.2694	0.4686
	sentiment	**0.2507**	**0.4682**
Speechiness	doc2vec	**0.0082**	**0.0619**
	sentiment	0.0124	0.082
Acousticness	doc2vec	0.0574	**0.1618**
	sentiment	**0.0488**	**0.1618**
Liveness	doc2vec	0.0256	**0.1122**
	sentiment	**0.0234**	0.1140
Danceability	doc2vec	**0.0270**	**0.1295**
	sentiment	0.0307	0.1416

Table 4.1: Evaluation results of several model, one for each Spotify feature, trained on Sentiment Representations and the Doc2Vec embeddings, that achieve the best value of MAE in the validation set. Values in bold identify the best performing model on the related feature.

Table 4.1 and Table 4.2 show the evaluated performances of our models in terms of MSE and MAE. Table 4.1 contains the results that we reached with both our semantic-based embeddings and the doc2vec representation with the model that achieved the lowest value of MAE on the validation set. Similarly, Table 4.2 shows the differences in performance of the models obtained with the lowest value of MSE on the validation set.

The results presented in the tables confirm our hypothesis about the target variables. The MSE smaller than 0.0018 for the Valence with the Sentiment-Based embeddings demonstrates that the model commits less consistent errors than the Doc2Vec approach. Regarding the Mode, we reach a proximity value for errors equal to 0.137 on average when using the sentiment embedding, reflecting improved results on the MAE. As fur-

Spotify Feature	Representation	MSE	MAE
Valence	doc2vec	0.0542	**0.1891**
	sentiment	**0.0524**	0.1911
Mode	doc2vec	0.2597	0.4803
	sentiment	**0.2470**	**0.4725**
Speechiness	doc2vec	**0.0082**	**0.0619**
	sentiment	0.0129	0.0741
Acousticness	doc2vec	**0.0508**	**0.1632**
	sentiment	0.0517	0.1907
Liveness	doc2vec	0.0256	**0.1122**
	sentiment	**0.0235**	0.1158
Danceability	doc2vec	**0.0268**	**0.1287**
	sentiment	0.0307	0.1423

Table 4.2: Evaluation results of several model, one for each Spotify feature, trained on Sentiment Representations and the Doc2Vec embeddings, that achieve the best value of MSE in the validation set. Values in bold identify the best performing model on the related feature.

ther proof of our intuition, the performances on Speechiness suggest that the semantic-based representation is more suitable for predicting targets that depend solely on textual features. The results for the other target variables are quite unclear, which depends on the fact that these features are not connected to the lyrics of a song but to the audio characteristics.

We also performed a T-Student test for comparing the predicted values with the actual targets, and in all the settings, we obtained a *p-value* ¡ 0.01. Therefore, the presented results are statistically reliable.

4.5 Conclusions

In this chapter, we have introduced a new unsupervised approach for extracting sentiment aspects from documents on a specific domain by detecting *Aspect Terms* and finding *Aspect Categories* to which they belong. Furthermore, we have formally defined *Aspect Term* and *Aspect Category* highlighting their differences, as well as the *Aspect-Based Sentiment Embedding*, a new kind of document representation based on sentiment values computed

for each *Aspect Category*. We have also provided an initial use case of our sentiment representation technique, motivating its benefits by performing an in-vitro experimental evaluation on Spotify's features regression task. We have compared our results with those obtained from semantic-based embeddings (i.e., Doc2Vec) on Spotify song lyrics, and we have demonstrated little improvements in the Mode and Valence features prediction.

To statistically validate our experiments, we have also performed a T-Student test on the predicted data that confirms the results reached. Despite the basic configuration of our method, we have proved that a sentiment representation could be more informative than a semantic one in a sentiment-related task.

5

Conversational Recommender

GUapp is a platform for job-postings search and recommendations for the Italian public administration. The platform offers recommendation services to match user skills and requests with job positions available in a given period. Based on Latent Dirichlet Allocation, the recommender system implemented in GUapp computes the k-nearest neighbor's job positions most similar to the user profile. Furthermore, to improve the user experience, GUapp implements a chatbot whose goal is to allow users to interact with the app through natural language. Thanks to that, the search and recommendation process becomes incremental, and the user can add new requirements at each stage of the interaction. In this chapter, we present GUapp, its recommender system, and the chatbot developed for achieving effective interaction with the user.

5.1 Introduction

Information overload is a well-known problem that impacts users' digital experience when they need to find interesting items in a large set of possible options [98]. Also, looking for a new job is a scenario when users particularly feel this problem. In this context, the only strategy for people is to manually declare job calls suitable or unsuitable for them, thus resulting in a clumsy user experience. Moreover, job calls usually have a limited time

for applying, so it is essential to constantly look for proper job openings over time as quickly as possible.

We developed the GUapp platform to find and discover job positions among job offers in the Italian public administration[1].

The information-retrieval (IR) and recommender-system (RS) research areas investigated this problem in the literature, where, in the first case, systems do not take into account the user's past preferences and retrieve the most relevant items according to the user query. On the other hand, although the goal of an RS and IR systems might seem similar, a user profile is built. It allows the proposal of services tailored to the user's specific characteristics and, in our case, to provide a *personalized* ranked list of items.

The GUapp's RS classifies jobs according to their topics and employs a k-Nearest Neighbor (k-NN) method to compute a personalized recommendation list. That list is updated daily and provided to the user. The user's past preferences identify a way to rank the job positions interesting to them. Moreover, GUapp offers a natural language-based interaction through a chatbot. The GUapp's chatbot allows the user to define her interests, describe her skills, and filter out results that do not match her requirements. Thus, GUapp must invoke different back-end services to satisfy the user's requests.

GUapp is available as a mobile app and a responsive Web client. The UI is dynamically adapted to the user behavior to show the correct view as the app opens proactively. Hence, the user's overall experience shows improvements by anticipating her needs. Hence, GUapp shows the list of new jobs, the recommendations, or the chatbot at startup, based on the user preferences.

5.2 The GUapp's Architecture

Figure 5.1 sketched the GUapp's architecture. We can find five main components: the Orchestrator, the Chatbot, the Recommender System, the User

[1] All the documents are freely available online at https://www.gazzettaufficiale.it/30giorni/concorsi

Profiler, and the Crawler.

The *Orchestrator* is a hub for GUapp. It manages the interaction between the different components of the system to satisfy the user's requests. Hence, for example, it invokes the recommender system when the user asks to receive a personalized list of job positions based on the information stored in her profile.

The *Chatbot* is the component of GUapp which allows users to interact through natural language powered by DialogFlow[2]. It performs two main tasks: intent and entity recognition. The *Intent Recognizer* (IR) analyzes the user's request expressed in natural language and understands her goal. That is a crucial step in identifying the back-end services to be invoked to accomplish the request. For example, if the user writes *I am looking for a new job in Bari*, the IR identifies as intent *new_job*. Actually, at this stage, GUapp is able to recognize three intents: *welcome*, *new_job*, *refine*.

The intent *welcome* is automatically recognized when the user activates the *chatbot*, which results in starting the conversation with a welcome message from the agent. The *new_job* intent is activated when the user sends a message looking for a new job. Finally, *refine* is an intent activated when the user adds new requirements to her previous request. For example, after the first message *I am looking for a new job in Bari*, the user might write *I prefer jobs at university*. In that case, *refine* adds this new request to the user request.

Once recognized the intent, the *Entity Recognizer* (ER) checks whether the sentence contains mentions of real-world entities. ER is implemented through DialogFlow as well. It adopts a fuzzy-matching strategy to identify entities in the user sentence. The set of entities is explicitly defined in a vocabulary. In the case the match succeeds, the recognized entity is returned. In the example above, the entity is *Bari*, a city in southern Italy. Now, the entities in the vocabulary are related only to cities and job-place categories (e.g., university, municipality).

The third component of the system is the *User Profiler* (UP). UP stores

[2]https://dialogflow.com/

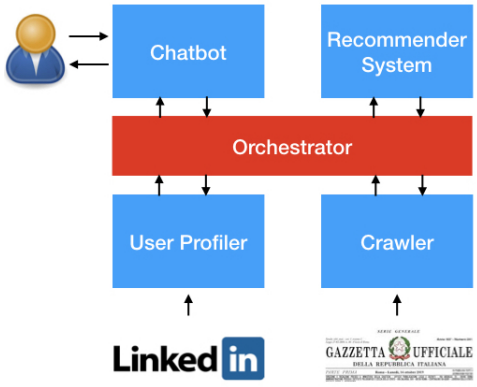


Figure 5.1: The GUapp's Architecture

and updates information about user preferences and habits. The user profile emerges by analyzing implicit and explicit feedback. As for the explicit ones, users can bookmark job positions and give "Like" feedback. Furthermore, analyzing natural language messages sent by the user to GUapp gathered further explicit feedback. Indeed, the user can *refine* the retrieved list of relevant job positions as described above. Other significant sources for getting user information are social networks, and more specifically, LinkedIn[3]. Indeed, GUapp offers authentication by LinkedIn APIs. Thanks to this integration, data from public information stored in the social network feed the user profile in GUapp. GUapp is able to get information about *reachable cities* and *skills*.

It is an effective strategy for addressing the *cold-start* problem. When the user signs in to the app for the first time, her profile is empty (i.e., *cold-start* situation). Thanks to the LinkedIn integration, GUapp can rank job calls by exploiting information stored in the social profile.

As for the implicit signals, we measure how long users read articles in the mobile app and from which view. Specifically, for each user, we record: i) a list of open positions, ii) searches, and iii) recommendations.

The *Recommender System* is another core component of GUapp, fully described in the following Section.

[3]https://www.linkedin.com/

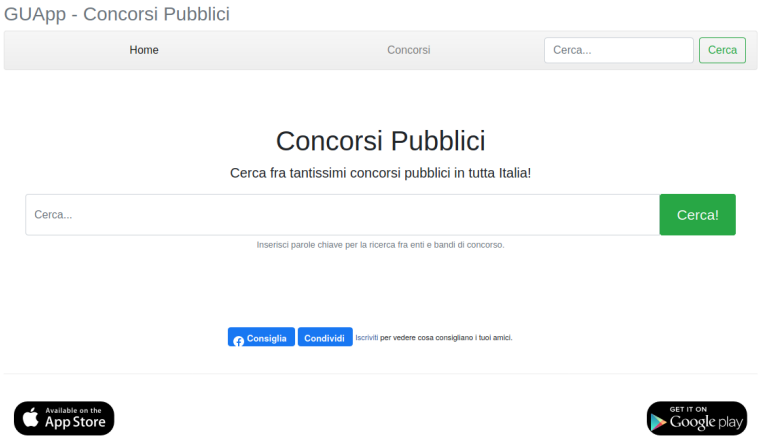


Figure 5.2: GUapp Web Site

Finally, GUapp implements a *Crawler* that daily extracts the job positions from *Gazzetta Ufficiale*, the official journal of record of the Italian government.

5.3 GUapp@work

As mentioned above, new job opportunities in public administration are announced daily. Our back-end infrastructure crawls new public calls daily. GUapp offers two kinds of UI: a Web Interface and an Android Application. Let us consider a running example to explain how GUapp works.

Paolo is looking for a new job, and he visits the GUapp Web Site[4]. Figure 5.2 shows the home page. Here Paolo types *Bari* in the search box, and he receives a list of open job positions located in Bari as shown in Figure 5.3. Calls are sorted in descending order of deadline, putting in the first positions those that expire first. Paolo browses the list of retrieved calls, but since the list is vast, he changes his query in the hope of better filtering the results. Since he would find a job position in a health institution, Paolo types as keywords *healthcare job calls in Bari*, which results in a more refined list of open job positions. Paolo finds an exciting job call; thus, he clicks on

[4]https://guapp.sisinflab.poliba.it

Details (i.e., Dettagli), and a new tab shows all the information published for that job position and a link to the official document provided by the recruiter.

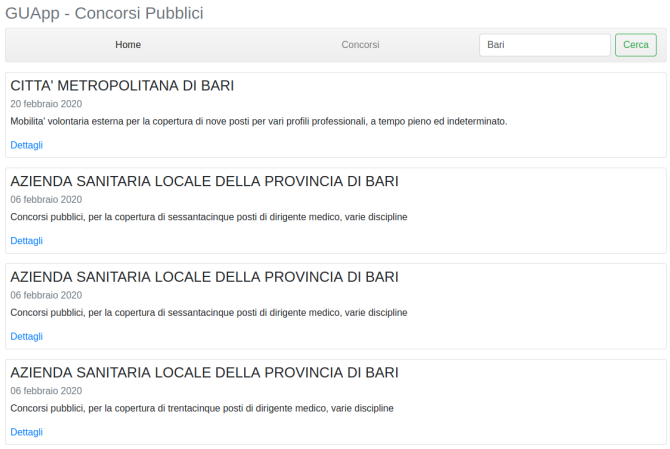
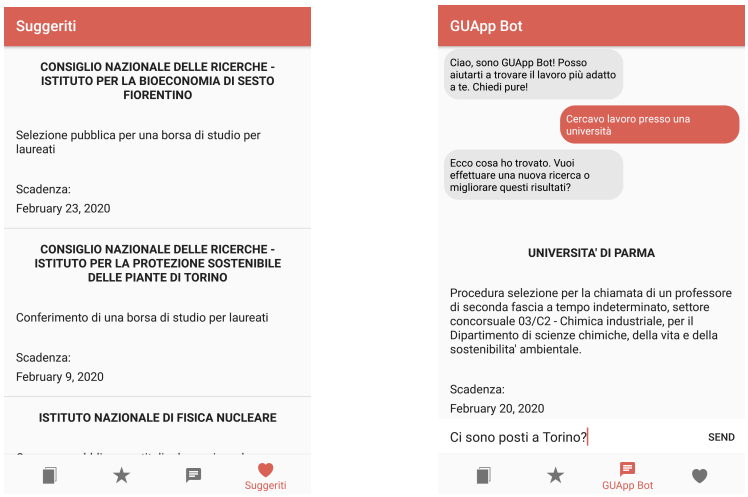


Figure 5.3: Searching results in `GUapp` Web Site

The details include the job place, type, application deadline, and a summary. However, the selected call only partially satisfies Paolo. Thus he explores the *Competitions* (i.e., Concorsi) tab, which contains all the available job calls provided by the *Gazzetta Ufficiale*. The `GUapp` Web Site can not help Paolo find his job since it is an open-job-position search engine, which leads Paolo to end his search hoping to find an interesting job call in the following days. Some days later, Paolo visits the `GUapp` Web Site again, but he has to start over his search since the website does not store his user profile and previous interactions.

Let us now consider the interaction through the Android app. The scenario is the same: Paolo is looking for a new job and launches the `GUapp` Android application on his smartphone. `GUapp` asks Paolo's credentials for the login. Indeed, the main difference between the mobile app and the website version is the capability of building a user profile. That is an excellent advantage for the user since he can receive personalized services. Paolo has two options: to create his account by providing an e-mail address and password or to sign in through the LinkedIn account.



(a) Recommendations (b) Chatbot

Figure 5.4: Screenshots of the Android application.

Paolo decides to sign in to the app through his LinkedIn account and chooses the information to share with GUapp. He grants to GUapp permissions for getting information on reachable locations and skills. Now, the app's primary view is a list of all job calls ordered by deadlines. Since Paolo does not want to browse a massive amount of items provided by the system, he selects the *Recommendations* (i.e., Suggeriti) view, which shows a list of recommended open job positions computed on his preferences, as depicted in Figure 5.4a. It is worth noting that the recommendation list is also pushed daily to Paolo proactively. The application exploits data gathered from LinkedIn to build the first list of recommendations. The list includes job calls whose location and requirements match Paolo's approachable cities and skills from LinkedIn. Here Paolo browses the recommended items that can potentially match his interests, and once he has found some exciting job calls, he taps them to get more details. Hence, GUapp shows more information for the selected job calls in a fashion remarkably similar to the web app. Paolo now expresses his preferences by tapping the corresponding like button for the calls deemed attractive. After Paolo rated all the open job positions he is interested he selects the *Favourite* (i.e., Favoriti)

view for marking the deadlines on his agenda.

GUapp implements also a conversational interface through a chatbot (Figure 5.4b). Accordingly, the user can dialog with GUapp, performing requests in natural language. Users can ask for a job position in a given location or say their preferences about a specific job opportunity by simply writing natural language sentences. The main difference compared to the search box is that the chatbot allows incrementally acquiring user preferences. Hence, Paolo, still looking for a job, selects the GUapp Bot view. GUapp Bot asks to exploit LinkedIn's information for the current session. However, Paolo ignores this option and sends the message *I am looking for a job at university*. Hence, GUapp shows a ranked list of job calls. The GUapp behavior is the same as the search box so far. However, Paolo can refine his request. He writes *Are their positions in Torino?*. At this point, GUapp has new information to refine the list of retrieved job positions. If Paolo wants to perform a new search, he sends the message *new search* and can start over.

5.4 The Document-Based Recommender System behind GUapp

The recommender system behind GUapp helps users find suitable job postings among daily published daily. GUapp recommends jobs by analyzing unstructured text that describes both requirements for the job application and job duties.

We first perform a pre-processing step on the text, removing stop-words from all the documents. Furthermore, words with a frequency greater than 95% and those that appear in the collection less than two times are removed, given their low informative power. The vocabulary size is limited to $1,000$ features (words). Thus, the final vocabulary $V = [w_1, \ldots, w_m]$ contains the m top words in the collection, ordered by their term frequency in the whole dataset. Each document d is represented in a Bag-of-Words matrix $R \in \mathbb{N}^{n \times m}$ where n denotes the number of documents, and m is the size of V. Each entry $r_{ij} \in R$ of the matrix is an integer counting the occurrences

for the term w_j in the document d_i.

Since we have no training data, the only option we can leverage to generate recommendations is to exploit *Unsupervised Learning* algorithms. We decided to use topic modeling to build clusters of words. Each document is then represented by all the topics t, each having a specific weight.

We adopted LDA [22] for topic modeling. It is a generative model that finds groups of words that frequently appear together across different documents. It allows linking text in a document to specific topics. LDA builds a *topic per document* Q and a *word per topic* P model, as shown in Figure 5.5. The algorithm has only one hyperparameter k representing the number of topics modeled. In GUapp, we empirically found that 10 is a good number of prior topics in our setting.

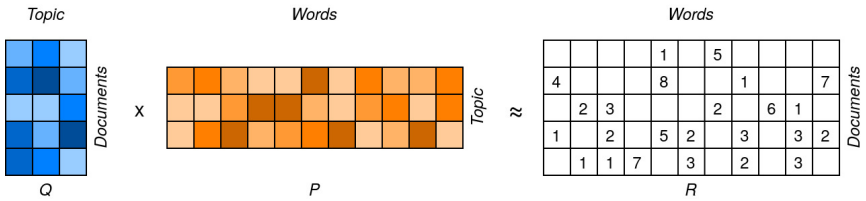


Figure 5.5: Topic modeling with Latent Dirichlet Allocation.

Given n documents, m words, and k prior number of topics, we train the model to predict: i) the distribution of words for each topic; ii) the distribution of topics for each document. LDA works by assigning a random topic t to each document word. Iteratively, for each word w and each document d, it computes $p(t \mid d)$ as the proportion of words in d currently assigned to t and $p(w \mid t)$ as the proportion of word w assigned to the topic t over the whole collection. Eventually, it reassigns each word w to a new topic, using the probability that topic t generates word w, as

$$t(w) = p(t \mid d) \cdot p(w \mid t)$$

In GUapp, we normalize the matrix Q, which represents the "document to topic" distribution so that topics for each row (document) range between 0 and 1. Since we want to recommend only open positions to which the

user might want to apply, we filter out all expired documents from R and compute the cosine similarity on the new matrix. For each topic, we select the top 50 documents never rated by the user and rank them according to their similarity score in the matrix. Records with a score equal to zero are filtered out. Thus, we generate a top-50 recommendation list of job positions for each user. It is noteworthy that, following [219], the topics of the recommended documents have the same proportion of the user's liked topics.

5.5 Conclusion

This chapter presented a platform for searching for jobs in Italian public administration. `GUapp` is composed of a recommender system that suggests relevant jobs to the users and a mobile app client that allows users to interact with the platform. One of the most exciting aspects of the mobile version is the chatbot integration that makes the interaction more natural. Indeed, preference elicitation becomes an incremental process, possibly refining and improving user requests. Although very useful from a practical perspective, the entire ecosystem results be an ideal candidate for AB tests related to user interactions and for the recommendation engine to be adopted in similar scenarios.

6

Knowledge Graphs for leading Dialogues

GUapp is an ecosystem for job-postings search and recommendation for the Italian public administration. Its main goal is to match user skills and requests with job positions available on the Gazzetta Ufficiale website, offering recommendation services in a conversational setting. We model Guapp's dialogues employing a domain-specific Knowledge Graph, which improves the users' natural language interaction with the app and the user experience. Thanks to that, the search and recommendation process becomes incremental, and the user can dynamically provide her preferences at each stage of the interaction. In this paper, we present GUapp and its overall architecture, besides the functioning of the conversational agent that dialogues with the user by exploiting a custom-designed Knowledge Graph. We also show a running example that outlines how GUapp models users and provides practical recommendations through natural language conversations.

6.1 Introduction

We stated in the previous Chapter how information overload affects the users' digital experience when selecting a suitable item among a large set

of alternatives, especially when searching for a new job. Moreover, job calls have a limited period for applying. For this reason, it is crucial to look for attractive job openings constantly. In this scenario, a system that only allows searching by a query composed of relevant keywords can make this task an ordeal for users.

GUapp offers a natural language-based interaction through a chatbot, allowing the user to define her interests, describe her skills, and filter out results that do not match her requirements. The system leverage the cold-start issue and the possible lack of items to suggest with a Conversational Agent. In addition, our system now interacts with users by exploiting a domain-specific Knowledge Graph (KG). The GUapp KG is obtained by merging some sub-graphs from state-of-the-art solutions like Dbpedia[1] and new triples generated from data scraped from external sources such as the ISTAT[2] website. From the latter, we have taken information about profession hierarchies and fields to which jobs belong.

Furthermore, we have built an ontology on which the retrieved facts rely. This KG allows our system to search for new semantically linked user preferences besides engaging in a negotiation phase when the proposed calls do not match all the user requirements. By exploiting the KG relations, GUapp can search for jobs that do not perfectly suit the user's preferences but remain close to her interests. On this line, conversations can reach a finer-grained level of detail on job aspects to recommend to users and enhance their expressiveness.

6.2 The architecture

Figure 6.1 sketches the renewed GUapp's architecture, composed of six main components: the Orchestrator, the Chatbot, the Recommender System, the User Profiler, the Crawler, and the Knowledge Graph.

The *Orchestrator* manages the interaction between the different components of the system. For example, it invokes the RS when the user asks to

[1]https://www.dbpedia.org/
[2]https://www.istat.it/

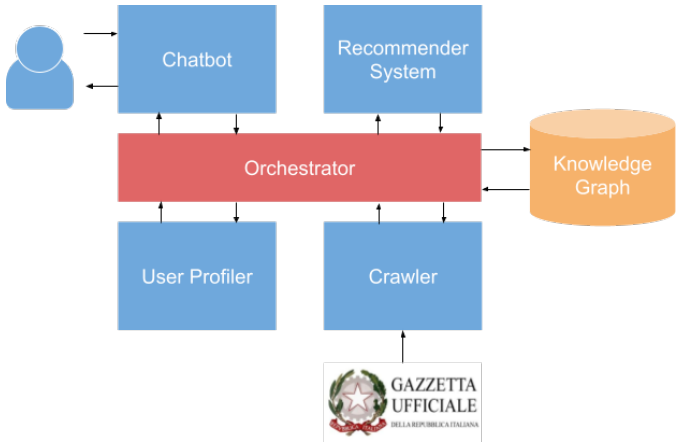


Figure 6.1: The GUapp's Architecture and its flow of data.

receive a list of job positions based on the information stored in her profile. In detail, this module leverages the overall data flow, selecting the appropriate component to solve specific tasks. We will consider the case where the Recommender System needs to start a negotiation phase with the user. The Orchestrator will collect data semantically related to her profile by querying the Knowledge Graph, exploited to find other items suitable to the user.

The *Chatbot* is the component of GUapp which allows users to interact through natural language, implemented by using DialogFlow[3], a Google platform for designing and integrating conversational user interfaces. For this purpose, the chatbot is equipped with an Intent Recognizer and an Entity Recognizer, allowing the system to understand several user requests and retrieve her essential data for the recommending task. That is a crucial step in identifying the back-end services to be invoked to accomplish the request. The*Intent Recognizer* analyzes the natural language request searching for specific goals such as collecting the user preferences, providing new job recommendations, or negotiating with the user. The *Entity Recognizer* checks whether the sentence contains mentions of real-world entities. It also exploits Dialogflow, and the KG entities power it. It adopts a fuzzy-

[3]https://dialogflow.com/

matching strategy to identify real-world entities in the user sentence. In the case the match succeeds, the recognized entity is returned.

The *User Profiler* instead collects all the users' preferences. In detail, we store all the data they provide during conversations, like favorite job locations, professions, etc. When the user signs into the app for the first time, her profile is empty (i.e., *cold-start* situation), but thanks to the chatbot, the user talks with `GUapp` about her skills, wishes, and ambitions. Then the system can rank job calls by exploiting the provided information. The User Profile is actively updated to the new user inputs, guaranteeing the recommendations be adaptive.

The *Recommender System* is another core component of `GUapp`. It exploits an Elasticsearch[4] index, which stores all the information scraped by the *Crawler* enriched with all the linked entities of our KG. In particular, for each job call, we automatically search for mentions of the KG entities or the ontological categories. The Elasticsearch documents will store the entity/category label related to each discovered mention with a confidence score, estimated with the BM25 algorithm, showing how reliable the labeling process outcomes are. The recommendations will be the job call closest to the user preferences with the highest confidence score.

The central intuition behind this model is to make dialogues as interactive and efficient as possible, allowing the system to negotiate with users whether results do not entirely match their preferences.

To be updated with all new jobs that the market offers, `GUapp` implements a *Crawler* that daily extracts the job positions from *Gazzetta Ufficiale*, the official journal of record of the Italian government. It directly communicates with the Orchestrator to store all the obtained data in the Elasticsearch instance as new documents.

Finally, we have provided the system a domain-specific *Knowledge Graph* built upon several sources, like Dbpedia and the ISTAT website, as stated before. The latter identifies a collection of raw data not in KG form. Consequently, we have modeled a new *Ontology* on which the overall KG can rely.

[4]https://www.elasticsearch.com

It defines relations and hierarchies about professions, competencies, locations, and so forth, and it wisely integrates the essential information with the already structured ones. It allows `GUapp` to be highly knowledgeable about job features crucial for the recommendation task. It also enables the chatbot to leverage fine-grained conversations and negotiations with users. To make the system more specialized in the job-opening domain, we plan to enrich our KG with further facts related to the job calls and positions recommendation.

6.3 Building the Knowledge Sources

The more detailed the features of items of a data collection, the higher the accuracy of the recommendations provided by the system. Following this intuition and given the conversational configuration of GUapp, we found owning a well-structured source of information an essential requisite. The system can provide more fine-grained recommendations using a knowledge source that defines aspects users evaluate to match their interests. For instance, job location and profession that a person could cover represent two main features that people consider while seeking a new job. At the same time, skills and experiences are crucial information to retrieve the most proper job position for the user. Employing a Knowledge Graph further allows the system to build semantically explicit user profiles. That results in highly interpretable recommendations, besides leading efficient negotiations in case no item satisfies all the user requirements.

On this line, we opted to enrich GUapp with a collection of Linked Open Data (LOD), suitable for the job recommendation task, since their availability in structured non-proprietary formats under an open license. Unfortunately, no KG and Ontologies are already available that outline the hierarchies of job professions and their belonging fields. Accordingly, we have started to implement a new LOD resource that perfectly fits the GUapp intents, besides being also available for other related purposes. To the best of our knowledge, the GUapp KG identifies the first attempt to structure relations between professions and their application fields under

the guidelines of the LOD protocols. Moreover, it also integrates all the data related to the job recommending task obtained from other state-of-the-art solutions, like cities, regions, and countries provided by Dbpedia.

We first collected all the RDF statements about locations from the Dbpedia project and the associated ontology to create a complete and consistent KG for this work. For this purpose, we have exploited the OpenLink Virtuoso[5], a Dbpedia SPARQL endpoint that allowed us to perform different SPARQL queries to collect the related data. Regarding professions and application fields, we opted to create a new ontology from scratch, assembling this information from highly reliable sources. The ISTAT website, managed by an Italian research institute for statistics, totally accomplishes this requirement. It stored a complete hierarchy of professions organized for sectors, application fields, and services in a tree data structure navigable through web pages for each position. For example, at the higher level, we can find distinctions between intellectual, technical, and office jobs; descending into the graph, we identify groups like scientific, health, and managing positions. This taxonomy reflects what the GUapp ontology asserts about professions. The leaves of the ISTAT tree describe all the jobs currently recognized in our society, like computer scientists and computer engineers, which compose some of the GUapp KG facts.

All these data are automatically retrieved from the previously mentioned website exploiting the Crawler routines. They not only collect all the job calls of the day, but they also scrape all the information that populates the KG. Then, we generate an owl file for the GUapp ontology, and all the RDF triples form the KG. For instance, the Computer Engineer profile belongs to the GUapp class Electronic Engineer, a subclass of Engineering, and the higher Intellectual and highly specialized Scientific profession class. These facts integrate those obtained from Dbpedia and form the overall GUapp KG. Currently, our KG is limited to the Italian language and has data restricted to professions, fields, and locations. Nevertheless, thanks to its semantic structure, we plan to expand it by adding several languages

[5]https://dbpedia.org/sparql/

besides including other job position features like user skills and goals.

Our system relies on this knowledge source mainly for implementing two functionalities. The first one is the labeling phase of the crawled dataset that makes the recommendation possible. In detail, the job calls retrieved by the Crawler are unstructured text, so we found it necessary to index the documents on a search engine like Elasticsearch. Looking for the KG labels in each job call, we enriched all the collected texts with the GUapp linked entities, which helps perform recommendations given the users' preferences. Instead, the second functionality allows the system to perform preference elicitation and negotiation steps during a conversation. Led by the GUapp KG entities and categories, our conversational agent realizes dialogues deeply related to the recommendation domain. It also grants to manage two highly felt issues in the RecSys community, like the cold start problem and the absence of items to suggest.

6.4 Recommending Jobs through Dialogues

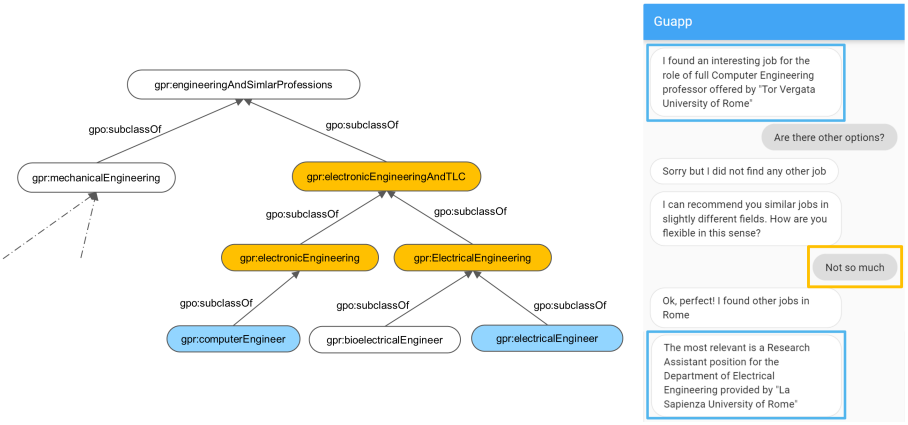


Figure 6.2: An example of Ontology driven negotiation. The messages in blue refer to the recommendations provided by the agent. The message in orange is the one that triggers the backward process in the ontology. Blue nodes in the ontology are the classes associated to the job proposals, while orange nodes are those explored in the negotiation phase.

This section presents an example of an ontology-driven conversation with details about how the knowledge base and the ontology handle the negotiation. Given the conversational nature of the system, a preliminary step for preference elicitation is needed. In this phase, the agent asks the user about her preferences. In particular, it asks about the geographical area and the field of interest. This first conversation phase follows a well-defined dialogue flow. Let us consider a scenario in which Claudio, a user who graduated as Computer Engineer, is looking for jobs in the computer engineering field. Therefore, he visits the GUapp website and finds the section related to the conversational agent. After a standard welcome message, the agent asks for the geographical area suitable for Claudio. In this case, Claudio writes that he is interested in working in Rome. The second crucial question that the agent asks Claudio is about the job position he would like to cover. He answers by stating that he is interested in a job as Computer Engineer. At this point, the agent tries to map Claudio's responses to entities and classes in the KG. In this particular case, the agent understands that Claudio is interested in a jobs proposal in Rome (linked to the entity *gpr:roma*) regarding a position of computer engineer (which refers to the ontological class *gpo:computerEngineer*). As a result, by exploiting the Elasticsearch index, the agent starts searching for jobs that match Claudio's requirements and creates a list of possible recommendations that perfectly match his preferences. The proposals are ranked based on the relevance score described in Section 6.2. Therefore, the agent provides only the first job proposals in the ranked list to not negatively impact the user experience. If Claudio asks for more results, the agent will explore the ranked list to provide other possible recommendations until he is satisfied or no more job proposal is available. If Claudio finds an exciting job call, the interaction ends. Otherwise, the scenario is more interesting since the agent cannot provide other solutions that perfectly match the user's needs. In that case, the agent needs more information to understand if the user is willing to travel or how flexible she is for the place. We refer to this phase as *negotiation*. Figure 6.2 shows an example of the agent's behavior in the negotiation and how it exploits the ontology for predicting other possible

recommendations. At the end of the preference elicitation phase shown in the previous example, the agent creates a ranked list of possible job recommendations in the computer engineering field in Rome. Thus it provides Claudio the first result, which is a job offer as a researcher offered by the University *La Sapienza* that is an instance of the *Computer Engineering* class in ontology. Let us assume that Claudio does not find the job proposal interesting and asks for more results. For the sake of simplicity, we assume that no more alternatives are available. To provide other solutions, the agent needs more information about Claudio. In particular, since there could be possible job offers in slightly different fields, it asks about the user's flexibility concerning the job place. Following the example, the agent asks Claudio *"I can recommend similar jobs in slightly different fields. How are you flexible in this sense?"*. The agent maps Claudio's answer to the number of edges it could navigate backward in the ontology. For instance, if Claudio answers *"Not so much"*, it means that he is not interested in jobs that differ too much from the field he proposed previously. For this reason, the answer is mapped to a maximum of 2 backward hops in the ontology. In case Claudio replies with *"Quite flexible"*, the agent maps the answer to three backward hops. This mapping has been empirically defined. Since our ontology is composed of 6 levels, two backward hops allow the system to provide job recommendations that belong to a similar domain as the previous ones. In this sense, the agents navigate backward the ontology starting from the class *gpo:computerEngineer* and reaching the parent node *gpo:electronicEngineeringAndTLC*. Starting from the inner node, reached after the backward phase, we get the leaves of the sub-tree with a forward step. Following the example, the agent reaches the leaf *gpo:electricalEngineer* that, for simplicity, is the only node in the current sub-tree associated with possible job offers in Rome. Also, in this case, it creates a ranked list following the same approach described previously and provides the user with the most relevant alternatives.

Another possible scenario in which the same method works is the following. Assuming that, during the negotiation phase, the user answers that she does not want the system to search for similar jobs or areas of work. In

this case, the agent will ask if the user is willing to travel, and according to the answer, it will provide job proposals more or less distant from the original request. Since the KB holds all the information about geographical entities, it allows the system to navigate the graph and find possible job offers in the same geographical area.

6.5 Conclusion

In this chapter, we presented `GUapp`, a platform searching for jobs in Italian public administration. We have outlined the architecture designed to make the job recommendation task possible in a conversational setting, besides describing the overall structure of the `GUapp` KG and its ontology. Thanks to that, `GUapp` consists of a recommender system that suggests relevant jobs to the users. One of the most exciting aspects is integrating a KG that helps drive the dialogue, making the interaction more natural and pushed at a finer-grained level. Indeed, the preference elicitation becomes incremental, possibly refining and improving the user requests.

7

Knowledge Graphs for Natural Language

The advent of pretrained language has renovated the ways of handling natural languages, improving the quality of systems that rely on them. BERT played a crucial role in revolutionizing the Natural Language Processing (NLP) area. However, the deep learning framework it implements lacks interpretability. Thus, recent research efforts aimed to explain what BERT learns from the text sources exploited to pre-train its linguistic model. This chapter analyzes the latent vector space resulting from the BERT context-aware word embeddings. We focus on assessing whether regions of the BERT vector space hold an explicit meaning attributable to a Knowledge Graph (KG). First, we demonstrate the existence of explicitly meaningful areas through the Link Prediction (LP) task. Then, we prove these regions are linked to explicit ontology concepts of a KG by learning classification patterns. This work represents the first attempt at interpreting the BERT learned linguistic knowledge through a KG relying on its pretrained context-aware word embeddings.

7.1 Introduction

Natural Language Processing (NLP) has experienced radical changes in its paradigms. Large amounts of linguistic data have promoted deep learning models to learn textual data representations at the expense of hand-crafted feature engineering approaches. This led to the design of highly successful architectures in implementing language models (i.e., Benjo et al. [17], and Mikolow et al. [150]). The growing interest of the research community, in parallel with the development of deep learning, has contributed to the explosion of a massive variety of NLP models to accomplish the most diverse applications [83]. The work of Vaswani et al. [232] stood out for its performance in solving the sequence transduction task with Transformers. Delvin et al. [52] got inspired by both this architecture and the gain in popularity of the pre-training and fine-tuning formula recorded in those years [153]. They proposed Bidirectional Encoder Representations from Transformers (BERT), which draw a turning point in the state-of-the-art NLP. Unlike the earlier proposed pre-trained language models [173, 87, 189], BERT implements a masked paradigm based on the Cloze task [224] and a next sentence prediction assignment to pre-train its language model. Such configuration, once fine-tuned, allowed BERT to achieve competitive performances on multiple benchmarks (e.g., GLUE [239], and SQuAD [195, 194]), which triggered the generation of many BERT variants to improve the resolution of the most popular NLP-based tasks.

Xia et al. [258] collected all the BERT implementations in five areas of progress. Four of them own the most published works on modifying BERT to reach specific goals, i.e., improving the language model acting on the pre-training objectives or data, the model efficiency, and the multilingual ability. In contrast, the fifth field on interpretability holds fewer works, reflecting the non-trivial task of interpreting such a sophisticated framework. Nevertheless, recent trends in published papers show an increasing interest in this topic. Many jobs inspect the model through attention heads [234, 38, 206, 95, 246], fine-tune it to solve interpretable tasks [121], or even modify the pre-training procedure for the same goals [225, 257].

On the other hand, only a minority poses questions on the semantic space conformations resulting from BERT. Some of them rely on classifiers to assess held information about the Entity-Linking [27], entity category clustering [45], and link prediction [176] tasks, which provide insights into the BERT learned knowledge. However, these solutions require authors to modify the embedding representations or target a multiclassification job. Thus, they are more proper to consider which information BERT learns to distinguish the embeddings rather than providing data on the properties of their space. Conversely, Ethayarajh [57] first studies the BERT latent space properties by comparing the cosine similarity between contextualized pre-trained BERT word embeddings. These word representations result from feeding BERT with words enclosed in contextual sentences without fine-tuning it. From this point, we will refer to this type of embedding simply as BERT embeddings. Ethayarajh states that BERT vector representations are anisotropic concerning their direction, forming groups in narrow cones. In this work, we aim to answer the following research questions:

- R1: Does BERT generate a latent semantic space holding information about knowledge with explicit semantics?
- R2: Can we learn functions to automatically detect precise knowledge graph concepts from the BERT latent space?

The study aims to investigate the BERT embedding space, looking for meaningful regions about explicit concepts and their edges. We rely on knowledge graphs (KGs) that connect entities via directed edges with exact semantics (relations), serialized via a set of triples subject-predicate-object where subject and objects represent unambiguous entities with a unique identifier. Given a KG, we first compare the behaviors of the BERT embeddings with several types of KG embeddings through Link Prediction (LP). Our intuition is that similarities between the two types of embeddings highlight that the BERT space contains the inherent structure of the KG. Also, they reflect the BERT embeddings holding explicit semantic information that depends on their position, thus areas that affect such information (i.e., a topology). We prove this property leads to exact KG concepts

(i.e., ontological classes). We learn patterns on BERT embeddings through several classifiers, one for each KG ontological category. Differently from a single classifier trained on a multi-classification task, individual binary classifiers extract feature patterns that uniquely identify each concept instead of checking BERT embedding differences for the assignment task. In other words, binary classifiers model regions in the BERT space that refer to the ontological categories. In contrast, the multi-classifier learns which features distinguish the word representations among the finite set of concepts, discarding the existence of other classes or the simultaneous belonging to multiple categories.

As the main contribution, we prove that BERT embeddings already possess information about the structure of a KG without needing any fine-tuning processes or architectural changes. We demonstrate that inner spatial properties of the BERT vector space allow inferring explicit KG concepts. Extensive experiments support our findings.

7.2 Methodology

The first goal we address is to prove the existence of explicit semantics behind the latent space resulting from BERT embeddings. BERT representations encode syntactic and hierarchical information (e.g., parts of speech, syntactic functions, subject-predicate agreements). Also, they are aware of semantic roles, entity types, and relations derived from linguistic knowledge [200]. However, the BERT vector space is mostly unexplored. We learn that the BERT space is anisotropic from [57]. Instead, Dalvi et al. [45] show that BERT embeddings can be grouped according to interpretable, but not explicit concepts. Thus, recent literature does not let us assert that the BERT space contains precise semantics.

Our approach analyzes similarities of the BERT embeddings with standard KG embeddings in solving the LP task. In detail, KG embeddings encode explicit semantics of a KG into a continuous vector space preserving its inherent structure [240]. At the same time, the LP focuses on predicting the correctness of unseen triplets (i.e., subject-predicate-object), which

also identifies a way to test the learning of a KG structure [228]. Thus, testing BERT embeddings with the LP will give further insights into the clear semantic information and the KG structure they may hold. To make our study significant, we lead our analysis under the following assumptions:

1. The BERT embeddings result from the BERT vanilla pre-trained model;

2. KG embeddings come out of training where both the subject and the object entities of a subject-predicate-object triple in a KG share the same vector space;

3. All the embeddings possess the same dimension.

Assumption (1) allows us to evaluate the intrinsic capability of BERT to encode explicit semantic information in its pre-trained language model, thus making deductions on its derived latent space. Fine-tuning procedures may specialize information held by BERT, affecting our study's generality. Therefore, we do not use variants like KG-BERT [266] or the one proposed by Petroni et al. [176] since they modify the pre-trained language model or fine-tune BERT. The last two assumptions make the comparisons between the BERT embeddings and the KG embeddings significant. On the one hand, with belief (2), we are interested in investigating only the vector space resulting from BERT; thus, KG embeddings and LP methods that require a space transformation or additional support spaces are out of our scope. On the other hand, constraint (3) places the examined embeddings on equal terms.

Given a KG, we compute the BERT embeddings on its entities. Since the most effective BERT representations in a semantic task are context-aware, we feed the BERT model with the entity label plus a related context. In particular, we compose sentences in the form of "label + *be* + abstract" for each KG entity. Then, we gain the contextualized BERT representation from the last hidden units that refers to the label. We do not make any assumptions about the granularity of the entity labels. Indeed, BERT operates on the WordPiece [255] segmentation of the input on a sub-word level, mainly to

avoid the mismatch vocabulary issue. Hence, each BERT hidden units represent single sub-words. The BERT embedding of the label results from aggregating its sub-word hidden units, which can belong to one or more words.

More precisely, let w be a word made of a set of WordPiece tokens t s.t. $w = [t_1, t_2, ..., t_n]$ and l be the entity label, which may have one o more words s.t. $l = [w_1, w_2, ...w_m]$. Thus, we refer with t_{ji} to the i-th WordPiece token of the j-th word, with $1 \leq i \leq n$ and $1 \leq j \leq m$. Given the BERT hidden unit h, the BERT embedding of a WordPiece token t results from $h_s(t)$, including the contextual information derived from the whole input sentence s. Therefore, $h_s(t_{ji})$ identifies the BERT embedding for the i-th WrodPiece token of the j-th word. In our approach, we denote with b the single word *be* and with c the set of d words composing the entity abstract. Thus, we can refer to the input we give to BERT for encoding each entity with $s = l + b + c$. Referring with w^l and t^l respectively to words and WordPiece tokens that belong to the entity label l, given an aggregating function f, the BERT embedding of an entity $\mathbf{e}$ derives from the aggregation of the WordPiece tokens embeddings of each word belonging to its label s.t. $\mathbf{e} = f(h_s(t^l_{ji}))$.

Once we encoded all the KG entities, we learned the relation embeddings over their entity BERT embeddings through existing LP models. We have chosen not to use BERT embeddings for relations as they differ from entities as semantic components. In an anisotropic space, such representations would depend on entities and relations directions. This way, we avoid assuming that relationships and entities have similar properties or analogous elements. To accomplish requirement (2), we opt for three well-established LP models such as TransE [25], TransH [243], and DistMult [262]. TransE is a translational distance model defining entities and relations as vectors in the same space. Given a KG fact (h, r, t), the relation is interpreted as a translation vector $\mathbf{r}$ so that the embedded entities $\mathbf{h}$ and $\mathbf{t}$ can be connected with $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Similarly, TransH follows the intuition of TransE by introducing relation-specific hyperplanes defined by the normal vector $\mathbf{w}_r$. Therefore, we have to first project the entity representations

$\mathbf{h}$ and $\mathbf{t}$ onto the hyperplane to have their connection: $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^T\mathbf{h}\mathbf{w}_r$, $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^T\mathbf{t}\mathbf{w}_r$. Instead, DistMult represents each relation as a diagonal matrix $\mathbf{M}_r = \text{diag}(\mathbf{r})$ modeling pairwise interactions between components of $\mathbf{h}$ and $\mathbf{t}$ along the same dimensions with a scoring function: $f_r(h,t) = \mathbf{h}^T\text{diag}(\mathbf{r})\mathbf{t}$. We apply these models directly to the BERT embeddings of the KG entities to infer their relations according to the KG structure. The BERT embeddings used for the entities remain fixed during the whole training process. Since the LP models encode the KG explicit semantics in the resulting KG embeddings, similarities of performances on the LP task for the KG embeddings and the BERT ones answer our research question R1 in Section 7.1.

The second objective is to assess whether the BERT space's properties enable the detection of precise KG concepts. To find those BERT space properties, we train several binary classifiers to detect if the BERT embedding of an entity belongs to an ontological class. We design each classifier to recognize a single concept, implementing one classifier for each KG class. In detail, this type of classifier will learn patterns from the BERT representations related to the concerned ontological concept. These classifiers know no other class than their own. Thus, they focus only on the input features (i.e., BERT embeddings) that allow deriving their class. These feature properties can then be extended to the whole embedding space. Reaching high test performances on each classifier answers to research question R2 in Section 7.1.

7.3 Semantic Analysis

This section provides the detailed configuration of the experiments that led to our analysis of the BERT space. We explore its inner properties through the embeddings computed by the pre-trained language model of the BERT-Base-cased. We investigate the cased version since we believe cased words enclose a different semantic than uncased ones.

We use Freebase (FB15k-237) [227] as the benchmark dataset to implement LP over the BERT embeddings. The central intuition here is that BERT

already contains Freebase explicit semantics since it was pre-trained on the English Wikipedia corpus that gives birth to Freebase. Since we need to feed BERT with sentences formed concatenating the label, the verb *be*, and the abstract for each entity, we discard from the FB15k-237 all the entities with no label or abstract in their Wikidata mapping[1]. Thus, we obtain an FB15K-237 subset by removing the facts related to the discarded entities, and we call it FB15K-237-Desc. It contains 266,263 facts over 13,667 entities compared to FB15K-237, which has 310,116 facts over 14,541. For completeness, we also compute separated entities' BERT representations by feeding the BERT model with only entity labels. These embeddings will benchmark the utility of the context for BERT in positioning them into the most appropriate space region. In both cases, the aggregation of the hidden units of the WordPiece tokens referring to the entities takes place through the arithmetic mean function.

7.3.1 Link Prediction and Semantics

Once we have computed the two BERT representations for all the FB15K-237-Desc entities, respectively BL (i.e., BERT Label) and BD (i.e., BERT Desc), we start the LP task in three configurations. The first one computes the standard KG embedding of TransE, TransH, and DistMult to compose our baselines. We need to recompute their performance to accomplish requirement (3) since their embedding size is 768. The second setting learns the relation embeddings over the BL entity representations, enabling the evaluation of the LP over the BL entity embeddings. The last scenario differs in the computation of the BERT entity embeddings through contextualized sentences (BD). All the configurations have their models trained on FB15K-237-Desc, split into 80%-10%-10% to generate the train, evaluation, and test sets. The training procedure follows the minibatch mode over the raw and filtered negative sampling proposed by Bordes et al. [25].

We used the grid search with early stopping to select the hyperparameters leading to the best performance in each configuration. The batch size

[1]https://developers.google.com/freebase/

	Raw			Filtered		
	MR	hit@10	hit@5	MR	hit@10	hit@5
TransE	**305.32**	**33.15**	**24.48**	**177.86**	**45.94**	**37.19**
TransH	412.73	29.74	21.39	271.88	40.98	32.39
DistMult	395.48	25.46	17.30	281.00	33.79	24.84
TransE_{BL}	866.47	21.30	15.88	773.43	25.03	19.56
TransH_{BL}	968.67	20.98	15.88	875.44	24.54	19.62
DistMult_{BL}	847.86*	19.84*	14.61*	753.46*	23.76*	18.09*
TransE_{BD}	604.83	<u>23.26</u>	<u>17.04</u>	508.40	<u>28.68</u>	<u>22.15</u>
TransH_{BD}	702.62	20.62	15.49	609.64	24.84	19.01
DistMult_{BD}	<u>560.64*</u>	21.31*	15.60*	<u>467.01*</u>	26.11*	19.83*

Table 7.1: Evaluation results of the LP task through the standard, BERT label (BL) and BERT description (BD) embeddings of TransE, TransH and DistMult. In bold, the best achieved results over the three configurations. The asterisks mark the BL and BD results closest to those of the standard model. The underlined outcomes highlight those values most relative to the best results.

has a value fixed to 1200 while the margin value γ can assume values among (1, 2, 10) for TransE, (0.25, 0.5, 1, 2) for TransH, and (0.001, 0.005, 0.01) for DistMult. In addition, TransH has its soft constraint weight selected among (0.015625, 0.0625, 0.25, 1), and TransE has its distance function tested between L1 and L2. TransE and TransH were trained with stochastic gradient descent (SGD) in the first LP configuration, while they exploited the Adam optimizer in the remaining two settings. DitsMult instead uses the AdaGrad optimizer in each LP scenario. We adopt the Mean Rank (MR) and the hit@n (i.e., hit@10 and hit@5) metrics to assess the LP performances. Table 7.1 resumes the results we achieved.

Surprisingly, the standard TransE reaches the best results in each configuration. This outcome derives from its known ability to catch the KG geometrical properties. In contrast, TransH and DistMult collect more semantic information, which can behave like noises given the different embedding dimensionality. The BL embeddings instead obtain the worst performances as we expected, further proving the importance of the context in correctly positioning the BERT representations in its vector space. The last configuration gives the most exciting results. Albeit maintaining the

same behaviors of the models, TransE over the BD embeddings gets comparable results with the standard DistMult model. This finding shows that we can infer meaningful explicit semantics by referring to the BERT embeddings' relevant properties (i.e., specific space dimensions). Therefore, we can say that the BERT space intrinsically contains a KG structure and precise semantics.

7.3.2 Ontological Analysis

The second experiment trains binary classifiers for each KG ontological class to detect whether a BERT embedding belongs to a specific category. Thus, we first retrieve from the Wikidata mapping all the entities' ontological classes through their *instanceOf* property, discarding those having no label or category. Since it contains more entities than its smaller counterpart, we use the FB15K [25] and limit our observations to those classes with enough entities to train meaningful classifiers. Hence, we obtain 20 categories with 9,258 entities distributed as in Figure 7.1.

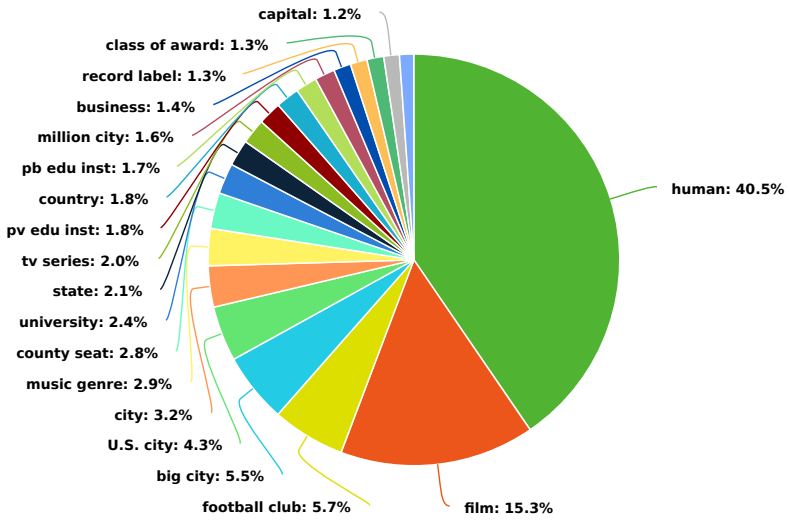


Figure 7.1: Distribution of the FB15K entities among the Wikidata ontological classes.

	Acc. (%)	P (%)	R (%)	F1 (%)	Supp. (#)
human	**99.95**	100	99.88	**99.94**	866
film	99.14	95.33	100	97.61	327
football club	99.78	96.82	100	98.39	122
big city	94.06	50.46	97.32	66.46	112
U.S. city	98.49	76.47	100	86.67	91
city	95.30	43.05	98.48	59.91	66
music genre	99.84	95.38	100	97.64	62
county seat	96.76	50	100	66.67	60
university	95.65	45.04	98.04	61.73	51
state	99.19	75.43	97.73	85.15	44
tv series	93.84	26.62	97.62	41.83	42
pv edu Inst	95.79	33.34	100	50	39
country	99.46	80	100	88.89	40
pb edu Inst	97.35	42.23	100	59.50	36
million city	94.60	23.08	100	37.50	30
business	95.63	26.36	100	41.73	23
record label	97.03	33.73	100	50.45	28
class of award	99.62	80	100	88.89	28
capital	94.55	18.70	95.83	31.29	24
o.a. publisher	95.46	21.70	95.84	35.38	24

Table 7.2: Evaluation results of the binary classifiers trained on each Wikidata category. In bold, the classifier's best results.

It is worth noting how classes like *city*, *big_city* and *US_city* can group into a single category. However, we first split the overall dataset into 80%-10%-10% to generate the train, validation, and test set. In this manner, we avoid classifiers knowing the entity class from the training set during their test. In addition, we perform this partition, maintaining the same distribution of classes. Then, for each category, we select from the train set all those samples that belong to the related classifier category. We added an equal amount of entities from other classes to have the train set balanced. We model each classifier as a feedforward neural network with a single hidden layer of 300 units that uses the ReLU activation function and the Adam optimizer. We evaluate the classifiers' performances through their accuracy, precision, recall, and F1 measure. Table 7.2 resumes the results of

each model and gives data about the positive output for each class through support.

As we can see, all the classifiers reach a high value of accuracy, among which the performance of the *human* classifier emerges. We explain these outcomes since the *human* classifier possesses most of the KG data supporting its modeling. Moreover, the *human* category identifies the most unambiguous class of the dataset, making it easy for its classifier to recognize its entities. Conversely, classes like *city*, *big_city*, and *US_city* have their classifier reaching low precision values despite their high recall. This result mainly depends on how these categories actually identify a single one. Indeed, the high recall highlights that the classifiers recognize all the positive samples of that class. At the same time, the low precision shows that they also identify other entities as belonging to that group. Therefore, We can infer that the BERT embeddings contain precise information that leads to explicit ontological classes, which can be extended to the spatial properties of the BERT vector space.

7.4 Conclusion

We have analyzed the latent vector space resulting from the BERT context-aware word embeddings, assessing whether the BERT vector space regions hold explicit semantics attributable to Knowledge Graphs (KGs). We have demonstrated the existence of explicitly meaningful areas exploiting the traits of the Link Prediction task. Then, we show these regions lead to explicit ontology concepts of a KG by learning classification patterns over the BERT embeddings. No previous works attempted interpreting the BERT learned linguistic knowledge through a Knowledge Graph.

8

Conclusion

This thesis presented a detailed analysis of Conversational Agents, focusing on their interactivity and elucidating the advantages that Knowledge Graphs convey to the conversation and understanding skills of such systems. This three-year work was characterized by an in-depth study of the vast panorama that describes the Conversational Agent, which allows us to propose innovative and practical solutions to the topic discussed so far.

In Chapter 3, we performed a literature review about Interactive Question Answering systems, which delivered fundamental concepts regarding Conversational Agents and the recently adopted implementations for high-performer interactive tools.

Secondly, in Chapter 4, we established a new approach based on Aspect-Based Sentiment Analysis that brought information gain to the sentiment-based application compared to the semantic one. Here, we proposed a sentiment-based text representation that enriched the interactivity of any systems relying on Natural Language Processing techniques.

We confirmed the usefulness of conversational techniques in Chapter 5 through the comparative study of a domain-oriented Recommender System, modeling the Conversational Recommender System GUapp.

We finally moved to study the potential offered by Knowledge Graphs in Chapters 6 and 7. Specifically, in Chapter 6, we equipped the Conversational Recommender System GUapp with a knowledge-aware dialogue

generator module, stating their efficiency in leading meaningful conversations for the job recommendation task.

Then, in Chapter 7, we assessed the intrinsic existence of a Knowledge Graph semantic in the BERT Masked Language Model. Our findings led to an improved generalization and understanding ability of systems that handle human languages, i.e., Conversational Agents.

Taken together, the research contributions presented in this dissertation enlightened new paths and methods that lead to general improvements of Conversational Agents, which are still far to be highly performant. We also outlined the limitations of our research findings that can inspire additional research in the topic aiming at more human-like and more intelligent chatbots. We hope that the content of this dissertation will serve as a beacon for future research in unveiling more accurate understanding and interaction abilities of Conversational Agents.

Bibliography

[1] Eleni Adamopoulou and Lefteris Moussiades. "Chatbots: History, technology, and applications". In: *Machine Learning with Applications* 2 (2020), p. 100006.

[2] Betty van Aken et al. "How Does BERT Answer Questions?: A Layer-Wise Analysis of Transformer Representations". In: *CIKM*. ACM, 2019, pp. 1823–1832.

[3] Kamran Alipour et al. "A Study on Multimodal and Interactive Explanations for Visual Question Answering". In: *SafeAI@AAAI*. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 54–62.

[4] Anna A Allen, Howard C Shane, and Ralf W Schlosser. "The Echo™ as a speaker-independent speech recognition device to support children with autism: an exploratory study". In: *Advances in Neurodevelopmental Disorders* 2.1 (2018), pp. 69–74.

[5] Francesca Alloatti, Luigi Di Caro, and Gianpiero Sportelli. "Real Life Application of a Question Answering System Using BERT Language Model". In: *SIGdial*. Association for Computational Linguistics, 2019, pp. 250–253.

[6] Pedro Álvarez et al. "DJ-Running: An Emotion-based System for Recommending Spotify Songs to Runners". In: *Proceedings of the 7th International Conference on Sport Sciences Research and Technology Support, icSPORTS 2019, Vienna, Austria, September 20-21, 2019*. Ed. by João Vilas-Boas, Pedro Pezarat Correia, and Jan Cabri. ScitePress, 2019, pp. 55–63.

[7] Flora Amato et al. "Classification of Web Job Advertisements: A Case Study". In: *23rd Italian Symposium on Advanced Database Systems, SEBD 2015, Gaeta, Italy, June 14-17, 2015*. 2015, pp. 144–151.

[8] Renzo Angles and Claudio Gutierrez. "Survey of graph database models". In: *ACM Computing Surveys (CSUR)* 40.1 (2008), pp. 1–39.

[9] Ashutosh Baheti, Alan Ritter, and Kevin Small. "Fluent Response Generation for Conversational Question Answering". In: *ACL*. Association for Computational Linguistics, 2020, pp. 191–207.

[10] Amir Bakarov, Vasiliy Yadrintsev, and Ilya Sochenkov. "Anomaly detection for short texts: Identifying whether your chatbot should switch from goal-oriented conversation to chit-chatting". In: *International Conference on Digital Transformation and Global Society*. Springer. 2018, pp. 289–298.

[11] Shivam Bansal, Aman Srivastava, and Anuja Arora. "Topic Modeling Driven Content Based Jobs Recommendation Engine for Recruitment Industry". In: *Procedia computer science* 122 (2017), pp. 865–872.

[12] Allan de Barcelos Silva et al. "Intelligent personal assistants: A systematic literature review". In: *Expert Systems with Applications* 147 (2020), p. 113193.

[13] Mohammad Ehsan Basiri et al. "ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis". In: *Future Gener. Comput. Syst.* 115 (2021), pp. 279–294.

[14] Kinjal Basu. "Conversational AI : Open Domain Question Answering and Commonsense Reasoning". In: *ICLP Technical Communications*. Vol. 306. EPTCS. 2019, pp. 396–402.

[15] Ranjan Kumar Behera et al. "Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data". In: *Inf. Process. Manag.* 58.1 (2021), p. 102435.

[16] Vito Bellini et al. "GUapp: a conversational agent for job recommendation for the Italian public administration". In: *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE. 2020, pp. 1–7.

[17] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model". In: *Advances in Neural Information Processing Systems* 13 (2000).

[18] Ana Berdasco et al. "User Experience Comparison of Intelligent Personal Assistants: Alexa, Google Assistant, Siri and Cortana". In: *International Conference on Ubiquitous Computing and Ambient Intelligence*. Vol. 31. MDPI Proceedings. MDPI, 2019, p. 51.

[19] Guillaume Le Berre and Philippe Langlais. "Attending Knowledge Facts with BERT-like Models in Question-Answering: Disappointing Results and Some Explanations". In: *Canadian Conference on AI*. Lecture Notes in Computer Science. Springer, 2020, pp. 356–367.

[20] Santanu Bhattacharjee et al. "Investigating Query Expansion and Coreference Resolution in Question Answering on BERT". In: *NLDB*. Lecture Notes in Computer Science. Springer, 2020, pp. 47–59.

[21] Eva AC Bittner, Sarah Oeste-Reiß, and Jan Marco Leimeister. "Where is the bot in our team? Toward a taxonomy of design option combinations for conversational agents in collaborative work". In: *Hawaii International Conference on System Sciences (HICSS)*. 2019.

[22] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3 (2003), pp. 993–1022. URL: http://jmlr.org/papers/v3/blei03a.html.

[23] Kurt Bollacker et al. “Freebase: a collaboratively created graph data base for structuring human knowledge”. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, pp. 1247–1250.

[24] Antoine Bordes, Y-Lan Boureau, and Jason Weston. “Learning end-to-end goal-oriented dialog”. In: *arXiv:1605.07683* (2016).

[25] Antoine Bordes et al. “Translating Embeddings for Modeling Multi-relational Data”. In: *NIPS*. 2013, pp. 2787–2795.

[26] Clara Bove, Jonathan Aigrain, and Marcin Detyniecki. “Building Trust in Artificial Conversational Agents.” In: *IUI Workshops*. 2021.

[27] Samuel Broscheit. “Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking”. In: *CoNLL*. Association for Computational Linguistics, 2019, pp. 677–685.

[28] Nil Goksel Canbek and Mehmet Emin Mutlu. “On the track of artificial intelligence: Learning with intelligent personal assistants”. In: *Journal of Human Sciences* 13.1 (2016), pp. 592–601.

[29] Jacky Casas et al. “Enhancing conversational agents with empathic abilities”. In: *Proceedings of the 21st ACM International Conference on Intelligent Virtual Agents*. 2021, pp. 41–47.

[30] Rakesh Chada. “Gendered Pronoun Resolution using BERT and an extractive question answering formulation”. In: *CoRR* 1906.03695 (2019).

[31] G. S. Chauhan et al. “A two-step hybrid unsupervised model with attention mechanism for aspect extraction”. In: *Expert Syst. Appl.* 161 (2020), p. 113673.

[32] Peng Chen et al. “Recurrent Attention Network on Memory for Aspect Sentiment Analysis”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Association for Computational Linguistics, 2017, pp. 452–461.

[33] Zhiyuan Chen et al. “Exploiting Domain Knowledge in Aspect Extraction”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2013, pp. 1655–1667.

[34] Ting-Rui Chiang, Hao-Tong Ye, and Yun-Nung Chen. “An Empirical Study of Content Understanding in Conversational Question Answering”. In: *AAAI*. AAAI Press, 2020, pp. 7578–7585.

[35] Jinhyuck Choi, Jin-Hee Song, and Yanggon Kim. “An Analysis of Music Lyrics by Measuring the Distance of Emotion and Sentiment”. In: *19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2018, Busan, Korea (South), June 27-29, 2018*. IEEE Computer Society, 2018, pp. 176–181.

[36] Philipp Christmann et al. “Look before you Hop: Conversational Question Answering over Knowledge Graphs Using Judicious Context Expansion”. In: *CIKM*. ACM, 2019, pp. 729–738.

[37] Kevin Clark and Christopher D. Manning. “Improving Coreference Resolution by Learning Entity-Level Distributed Representations”. In: *Association for Computational Linguistics (ACL)*. 2016.

[38] Kevin Clark et al. “What Does BERT Look at? An Analysis of BERT’s Attention”. In: *BlackboxNLP@ACL*. Association for Computational Linguistics, 2019, pp. 276–286.

[39] Christopher Clarke et al. “One Agent To Rule Them All: Towards Multi-agent Conversational AI”. In: *arXiv preprint arXiv:2203.07665* (2022).

[40] Chloe Clavel and Zoraida Callejas. “Sentiment analysis: from opinion mining to human-agent interaction”. In: *IEEE Transactions on affective computing* 7.1 (2015), pp. 74–93.

[41] Ron Cole et al. “New tools for interactive speech and language training: using animated conversational agents in the classroom of profoundly deaf children”. In: *MATISSE-ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education*. 1999.

[42] L. Cui et al. “Superagent: A customer service chatbot for e-commerce websites”. In: *Proceedings of ACL 2017, system demonstrations*. 2017, pp. 97–102.

[43] Sérgio Curto et al. “JUST.ASK, a QA system that learns to answer new questions from previous interactions”. In: *LREC*. European Language Resources Association (ELRA), 2014, pp. 2603–2607.

[44] Luis Fernando D’Haro, Seokhwan Kim, and Rafael E Banchs. “A robust spoken Q&A system with scarce in-domain resources”. In: *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE. 2015, pp. 47–53.

[45] Fahim Dalvi et al. “Discovering Latent Concepts Learned in BERT”. In: *CoRR* abs/2205.07237 (2022).

[46] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. “Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-Based Lookup through the User Interaction”. In: *ESWC (1)*. Vol. 6088. Lecture Notes in Computer Science. Springer, 2010, pp. 106–120.

[47] Rajarshi Das et al. “Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering”. In: *ICLR (Poster)*. OpenReview.net, 2019.

[48] Toon De Pessemier, Kris Vanhecke, and Luc Martens. “A scalable, high-performance Algorithm for hybrid job recommendations”. In: *Proceedings of the Recommender Systems Challenge*. 2016, pp. 1–4.

[49] Clément Delgrange, Jean-Michel Dussoux, and Peter Ford Dominey. “Usage-based learning in human interaction with an adaptive virtual assistant”. In: *IEEE Transactions on Cognitive and Developmental Systems* 12.1 (2019), pp. 109–123.

[50] Kerstin Denecke et al. “Self-anamnesis with a conversational user interface: concept and usability study”. In: *Methods of information in medicine* 57.05/06 (2018), pp. 243–252.

[51] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186.

[52] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv:1810.04805* (2018).

[53] Stephan Diederich, Alfred Benedikt Brendel, and Lutz M Kolbe. “Designing anthropomorphic enterprise conversational agents”. In: *Business & Information Systems Engineering* 62.3 (2020), pp. 193–209.

[54] Tuong Do et al. “Compact Trilinear Interaction for Visual Question Answering”. In: *ICCV*. IEEE, 2019, pp. 392–401.

[55] Jesse Dodge et al. “Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems”. In: *arXiv:1511.06931 [cs]* (Nov. 2015). URL: `http://arxiv.org/abs/1511.06931` (visited on 06/19/2019).

[56] Jesse Dodge et al. “Evaluating prerequisite qualities for learning end-to-end dialog systems”. In: *arXiv preprint arXiv:1511.06931* (2015).

[57] Kawin Ethayarajh. “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 55–65.

[58] Oren Etzioni. “Search needs a shake-up”. In: *Nat.* 476.7358 (2011), pp. 25–26.

[59] Ahmed Fadhil. “A conversational Interface to improve medication adherence: Towards AI support in Patient’s treatment”. In: *arXiv preprint arXiv:1803.09844* (2018).

[60] Ahmed Fadhil and Silvia Gabrielli. "Addressing challenges in promoting healthy lifestyles: the al-chatbot approach". In: *Proceedings of the 11th EAI international conference on pervasive computing technologies for healthcare*. 2017, pp. 261–265.

[61] Ronald Fagin, Amnon Lotem, and Moni Naor. "Optimal aggregation algorithms for middleware". In: *Journal of computer and system sciences* 66.4 (2003), pp. 614–656.

[62] Ethan Fast et al. "Iris: A conversational agent for complex tasks". In: *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018, pp. 1–12.

[63] Niroshinie Fernando et al. "Examining digital assisted living: Towards a case study of smart homes for the elderly". In: (2016).

[64] Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. "Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (Woebot): a randomized controlled trial". In: *JMIR mental health* 4.2 (2017), e7785.

[65] Jun-ichi Fukumoto, Noriaki Aburai, and Ryosuke Yamanishi. "Interactive Document Expansion for Answer Extraction of Question Answering System". In: *KES*. Vol. 22. Procedia Computer Science. Elsevier, 2013, pp. 991–1000.

[66] Russell Fulmer et al. "Using psychological artificial intelligence (Tess) to relieve symptoms of depression and anxiety: randomized controlled trial". In: *JMIR mental health* 5.4 (2018), e9782.

[67] Peng Gao et al. "Multi-Modality Latent Interaction Network for Visual Question Answering". In: *ICCV*. IEEE, 2019, pp. 5824–5834.

[68] Lorenzo Gatti, Marco Guerini, and Marco Turchi. "SentiWords: Deriving a High Precision and High Coverage Lexicon for Sentiment Analysis". In: *IEEE Trans. Affect. Comput.* 7.4 (2016), pp. 409–421.

[69] Athanasios Giannakopoulos et al. "Unsupervised Aspect Term Extraction with B-LSTM & CRF using Automatically Labelled Datasets". In: *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*. Ed. by Alexandra Balahur, Saif M. Mohammad, and Erik van der Goot. Association for Computational Linguistics, 2017, pp. 180–188.

[70] Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. "Towards Designing Cooperative and Social Conversational Agents for Customer Service." In: *ICIS*. 2017.

[71] Hitesh Golchha et al. "Courteously yours: Inducing courteous behavior in customer care responses using reinforced pointer generator network". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 851–860.

[72] Carlos A Gomez-Uribe and Neil Hunt. "The netflix recommender system: Algorithms, business value, and innovation". In: *ACM Transactions on Management Information Systems (TMIS)* 6.4 (2015), pp. 1–19.

[73] Daniel Gordon et al. "IQA: Visual Question Answering in Interactive Environments". In: *CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4089–4098.

[74] Daya Guo et al. "Dialog-to-Action: Conversational Question Answering Over a Large-Scale Knowledge Base". In: *NeurIPS 2018*. 2018, pp. 2946–2955.

[75] Maryam Habibi, Parvaz Mahdabi, and Andrei Popescu-Belis. "Question answering in conversations: Query refinement using contextual and semantic information". In: *Data Knowl. Eng.* 106 (2016), pp. 38–51.

[76] Hojae Han et al. “MICRON: Multigranular Interaction for Contextualizing RepresentatiON in Non-factoid Question Answering”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 5889–5894.

[77] Abdalraouf Hassan and Ausif Mahmood. “Convolutional Recurrent Deep Learning Model for Sentence Classification”. In: *IEEE Access* 6 (2018), pp. 13949–13957.

[78] Steven Haussmann et al. “FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation”. In: *ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. 2019, pp. 146–162.

[79] Johann Hauswald et al. “Designing future warehouse-scale computers for sirius, an end-to-end voice and vision personal assistant”. In: *ACM Transactions on Computer Systems (TOCS)* 34.1 (2016), pp. 1–32.

[80] Sofian Hazrina et al. “Review on the advancements of disambiguation in semantic question answering system”. In: *Inf. Process. Manag.* 53.1 (2017), pp. 52–69.

[81] Ruidan He et al. “An Unsupervised Neural Attention Model for Aspect Extraction”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Ed. by Regina Barzilay and Min-Yen Kan. Association for Computational Linguistics, 2017, pp. 388–397.

[82] Shizhu He, Kang Liu, and Weiting An. “Learning to Align Question and Answer Utterances in Customer Service Conversation with Recurrent Pointer Networks”. In: *AAAI*. AAAI Press, 2019, pp. 134–141.

[83] Julia Hirschberg and Christopher D Manning. “Advances in natural language processing”. In: *Science* 349.6245 (2015), pp. 261–266.

[84] Lynette Hirschman and Robert J. Gaizauskas. "Natural language question answering: the view from here". In: *Nat. Lang. Eng.* 7.4 (2001), pp. 275–300.

[85] Aidan Hogan et al. "Knowledge graphs". In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–37.

[86] Yining Hong et al. "Academic Reader: An Interactive Question Answering System on Academic Literatures". In: *AAAI*. AAAI Press, 2019, pp. 9855–9856.

[87] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *ACL (1)*. Association for Computational Linguistics, 2018, pp. 328–339.

[88] Chun Hsiang Hsieh and Daniel J Buehrer. "The implementation of an artificially intelligent personal assistant for a personal computer". In: *Applied Mechanics and Materials*. Vol. 627. Trans Tech Publ. 2014, pp. 372–376.

[89] Jun Hu et al. "Attentive Interactive Convolutional Matching for Community Question Answering in Social Multimedia". In: *ACM Multimedia*. ACM, 2018, pp. 456–464.

[90] Minqing Hu and Bing Liu. "Mining and summarizing customer reviews". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*. Ed. by Won Kim et al. ACM, 2004, pp. 168–177.

[91] Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. "Flowqa: Grasping flow in history for conversational machine comprehension". In: *arXiv preprint arXiv:1810.06683* (2018).

[92] Eric Hulburd. "Exploring BERT Parameter Efficiency on the Stanford Question Answering Dataset v2.0". In: *CoRR* abs/2002.10670 (2020).

[93] Shafquat Hussain, Omid Ameri Sianaki, and Nedal Ababneh. "A survey on conversational agents/chatbots classification and design techniques". In: *Workshops of the International Conference on Advanced Information Networking and Applications*. Springer. 2019, pp. 946–956.

[94] Andrea Iovine, Fedelucio Narducci, and Giovanni Semeraro. "Conversational Recommender Systems and Natural Language: A study through the ConveRSE Framework". In: *Decision Support Systems* (2020), p. 113250. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2020.113250.

[95] Sarthak Jain and Byron C. Wallace. "Attention is not Explanation". In: *NAACL-HLT (1)*. Association for Computational Linguistics, 2019, pp. 3543–3556.

[96] Niklas Jakob and Iryna Gurevych. "Extracting Opinion Targets in a Single and Cross-Domain Setting with Conditional Random Fields". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2010, pp. 1035–1045.

[97] Dietmar Jannach et al. "A survey on conversational recommender systems". In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–36.

[98] Dietmar Jannach et al. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[99] Weike Jin et al. "Multi-interaction Network with Object Relation for Video Question Answering". In: *ACM Multimedia*. ACM, 2019, pp. 1193–1201.

[100] Rie Johnson and Tong Zhang. "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks". In: *NAACL*. 2015.

[101] Ying Ju et al. "Technical report on Conversational Question Answering". In: *CoRR* abs/1909.10772 (2019).

[102] Michael Jugovac and Dietmar Jannach. "Interacting with Recommenders: Overview and Research Directions". In: *ACM Trans. Interact. Intell. Syst.* 7.3 (Sept. 2017), 10:1–10:46. ISSN: 2160-6455.

[103] Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. "Conversational Question Answering over Passages by Leveraging Word Proximity Networks". In: *SIGIR*. ACM, 2020, pp. 2129–2132.

[104] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A Convolutional Neural Network for Modelling Sentences". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 655–665.

[105] Seokhwan Kim and Rafael E Banchs. "R-cube: a dialogue agent for restaurant recommendation and reservation". In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*. IEEE. 2014, pp. 1–6.

[106] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751.

[107] Robin Knote et al. "The what and how of smart personal assistants: principles and application domains for IS research". In: *Multikonferenz Wirtschaftsinformatik (MKWI)* (2018).

[108] Ned Kock. "The psychobiological model: Towards a new theory of computer-mediated communication based on Darwinian evolution". In: *Organization science* 15.3 (2004), pp. 327–348.

[109] Artemy Kolchinsky et al. "The Minor Fall, the Major Lift: Inferring Emotional Valence of Musical Chords through Lyrics". In: *CoRR* abs/1706.08609 (2017). arXiv: 1706.08609.

[110] Natalia Konstantinova and Constantin Orasan. "Interactive question answering". In: *Emerging applications of natural language processing: concepts and new research*. IGI Global, 2013, pp. 149–169.

[111] Mayank Kulkarni and Kristy Elizabeth Boyer. "Toward Data-Driven Tutorial Question Answering with Deep Learning Conversational Models". In: *BEA@NAACL-HLT*. Association for Computational Linguistics, 2018, pp. 273–283.

[112] Girish Kumar et al. "Question-Answer Selection in User to User Marketplace Conversations". In: *IWSDS*. Vol. 579. Lecture Notes in Electrical Engineering. Springer, 2018, pp. 397–403.

[113] M Naveen Kumar et al. "Android based educational Chatbot for visually impaired people". In: *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE. 2016, pp. 1–4.

[114] Souvik Kundu, Qian Lin, and Hwee Tou Ng. "Learning to Identify Follow-Up Questions in Conversational Question Answering". In: *ACL*. Association for Computational Linguistics, 2020, pp. 959–968.

[115] Chia-Chih Kuo, Shang-Bao Luo, and Kuan-Yu Chen. "An Audio-Enriched BERT-Based Framework for Spoken Multiple-Choice Question Answering". In: *INTERSPEECH*. ISCA, 2020, pp. 4173–4177.

[116] Veronica Latcinnik and Jonathan Berant. "Explaining Question Answering Models through Text Generation". In: *CoRR* abs/2004.05569 (2020).

[117] Quoc V. Le and Tomás Mikolov. "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. JMLR Workshop and Conference Proceedings. 2014, pp. 1188–1196.

[118] Changyoon Lee et al. "automaTA: Human-Machine Interaction for Answering Context-Specific Questions". In: *L@S*. ACM, 2019, 44:1–44:4.

[119] Danielle H Lee and Peter Brusilovsky. "Fighting information overflow with personalized comprehensive information access: A proac-

tive job recommender". In: *Third International Conference on Autonomic and Autonomous Systems (ICAS'07)*. IEEE. 2007, pp. 21–21.

[120] Zijian Lew et al. "Interactivity in online chat: Conversational contingency and response latency in computer-mediated communication". In: *Journal of Computer-Mediated Communication* 23.4 (2018), pp. 201–221.

[121] Da Li et al. "Multi-task Pre-training Language Model for Semantic Network Completion". In: *arXiv preprint arXiv:2201.04843* (2022).

[122] Fei Li and H. V. Jagadish. "Constructing an Interactive Natural Language Interface for Relational Databases". In: *Proc. VLDB Endow.* 8.1 (2014), pp. 73–84.

[123] Huayu Li et al. "A Context-aware Attention Network for Interactive Question Answering". In: *KDD*. ACM, 2017, pp. 927–935.

[124] Qian Li et al. "Answer-Supervised Question Reformulation for Enhancing Conversational Machine Comprehension". In: *MRQA*. Association for Computational Linguistics, 2019, pp. 38–47.

[125] Qing Li et al. "Tell-and-Answer: Towards Explainable Visual Question Answering using Attributes and Captions". In: *EMNLP*. Association for Computational Linguistics, 2018, pp. 1338–1346.

[126] Qing Li et al. "VQA-E: Explaining, Elaborating, and Enhancing Your Answers for Visual Questions". In: *ECCV (7)*. Vol. 11211. Lecture Notes in Computer Science. Springer, 2018, pp. 570–586.

[127] Ronghan Li et al. "Directional attention weaving for text-grounded conversational question answering". In: *Neurocomputing* 391 (2020), pp. 13–24.

[128] Xin Li and Wai Lam. "Deep Multi-Task Learning for Aspect Term Extraction with Memory Interaction". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics, 2017, pp. 2886–2892.

[129] Erin Chao Ling et al. “Factors influencing users’ adoption and use of conversational agents: A systematic review”. In: *Psychology & Marketing* 38.7 (2021), pp. 1031–1051.

[130] Aiting Liu et al. “BB-KBQA: BERT-Based Knowledge Base Question Answering”. In: *CCL*. Vol. 11856. Lecture Notes in Computer Science. Springer, 2019, pp. 81–92.

[131] Bing Liu and Ian Lane. “An End-to-End Trainable Neural Network Model with Belief Tracking for Task-Oriented Dialog”. en. In: *Interspeech 2017* (Aug. 2017). arXiv: 1708.05956, pp. 2506–2510. DOI: 10.21437/Interspeech.2017-1326. URL: http://arxiv.org/abs/1708.05956 (visited on 03/21/2019).

[132] Qian Liu et al. “Automated Rule Selection for Aspect Extraction in Opinion Mining”. In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. Ed. by Qiang Yang and Michael J. Wooldridge. AAAI Press, 2015, pp. 1291–1297.

[133] Qian Liu et al. “Improving Opinion Aspect Extraction Using Semantic Similarity and Aspect Associations”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 2986–2992.

[134] Siyuan Liu et al. “NEURON: An Interactive Natural Language Interface for Understanding Query Execution Plans in RDBMS”. In: *CoRR* abs/1805.05670 (2018).

[135] Song Liu, Yixin Zhong, and Fuji Ren. “Interactive Question Answering Based on FAQ”. In: *CCL*. Vol. 8202. Lecture Notes in Computer Science. Springer, 2013, pp. 73–84.

[136] James Lockett et al. “Intelligent Voice Agent and Service (iVAS) for Interactive and Multimodal Question and Answers”. In: *FQAS*. Lecture Notes in Computer Science. Springer, 2019, pp. 396–402.

[137] Huaishao Luo et al. “Improving Aspect Term Extraction With Bidirectional Dependency Tree Representation”. In: *IEEE ACM Trans. Audio Speech Lang. Process.* 27.7 (2019), pp. 1201–1212.

[138] Quim M., Xavier F., and Jordi M. “Conversational Agents in Software Engineering: Survey, Taxonomy and Challenges”. In: *arXiv* (2021).

[139] Yoelle Maarek. “Alexa and Her Shopping Journey”. In: *CIKM*. ACM, 2018, p. 1.

[140] J. MacQueen. “Some Methods for Classification and Analysis of Multivariate Observations”. In: *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*. Ed. by L. M. Le Cam and J. Neyman. University of California Press, Berkeley, CA, USA, 1967, pp. 281–297.

[141] Anutosh Maitra, Shivam Garg, and Shubhashis Sengupta. “Enabling Interactive Answering of Procedural Questions”. In: *NLDB*. Lecture Notes in Computer Science. Springer, 2020, pp. 73–81.

[142] Jochen Malinowski et al. “Matching people and jobs: A bilateral recommendation approach”. In: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS’06)*. Vol. 6. IEEE. 2006, pp. 137c–137c.

[143] Angrosh Mandya, Danushka Bollegala, and Frans Coenen. “Evaluating Co-reference Chains Based Conversation History in Conversational Question Answering”. In: *PACLING*. Vol. 1215. Communications in Computer and Information Science. Springer, 2019, pp. 280–292.

[144] Angrosh Mandya et al. “Do not let the history haunt you: Mitigating Compounding Errors in Conversational Question Answering”. In: *LREC*. European Language Resources Association, 2020, pp. 2017–2025.

[145] Yosi Mass et al. “A Study of BERT for Non-Factoid Question Answering under Passage Length Constraints”. In: *CoRR* 1908.06780 (2019).

[146] J. S. McCarley. “Pruning a BERT-based Question Answering Model”. In: *CoRR* abs/1910.06360 (2019).

[147] Graeme McLean and Kofi Osei-Frimpong. “Hey Alexa… examine the variables influencing the use of artificial intelligent in-home voice assistants”. In: *Computers in Human Behavior* 99 (2019), pp. 28–37.

[148] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. “Sentiment analysis algorithms and applications: A survey”. In: *Ain Shams engineering journal* 5.4 (2014), pp. 1093–1113.

[149] Pablo N Mendes, Max Jakob, and Christian Bizer. *DBpedia: A multilingual cross-domain knowledge base*. European Language Resources Association (ELRA), 2012.

[150] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[151] Tomás Mikolov, Quoc V. Le, and Ilya Sutskever. “Exploiting Similarities among Languages for Machine Translation”. In: (2013). arXiv: 1309.4168.

[152] Martijn Millecamp et al. “Controlling Spotify Recommendations: Effects of Personal Characteristics on Music Recommender User Interfaces”. In: *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*. Ed. by Tanja Mitrovic et al. ACM, 2018, pp. 101–109.

[153] Bonan Min et al. “Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey”. In: *arXiv preprint arXiv:2111.01243* (2021).

[154] Seungwhan Moon et al. “Memory Graph Networks for Explainable Memory-grounded Question Answering”. In: *CoNLL*. Association for Computational Linguistics, 2019, pp. 728–736.

[155] Subhabrata Mukherjee and Sachindra Joshi. “Help yourself: a virtual self-assist system”. In: *Proceedings of the 23rd International Conference on World Wide Web*. 2014, pp. 171–174.

[156] Thomas Müller et al. "Answering Conversational Questions on Structured Data without Logical Forms". In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 5901–5909.

[157] Florian N. and Wolfgang M. "Justification and transparency explanations in dialogue systems to maintain human-computer trust". In: *Situated Dialog in Speech-Based Human-Computer Interaction*. Springer, 2016, pp. 41–50.

[158] Swen Nadkarni and Reinhard Prügl. "Digital transformation: a review, synthesis and opportunities for future research". In: *Management Review Quarterly* 71.2 (2021), pp. 233–341.

[159] M. Asif Naeem, Saif Ullah, and Imran Sarwar Bajwa. "Interacting with Data Warehouse by Using a Natural Language Interface". In: *NLDB*. Vol. 7337. Lecture Notes in Computer Science. Springer, 2012, pp. 372–377.

[160] Fedelucio Narducci et al. "An investigation on the user interaction modes of conversational recommender systems for the music domain". In: *User Modeling and User-Adapted Interaction* (2019), pp. 1–34. DOI: https://doi.org/10.1007/s11257-019-09250-7.

[161] Tetsuya Nasukawa and Jeonghee Yi. "Sentiment analysis: capturing favorability using natural language processing". In: *Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP 2003), October 23-25, 2003, Sanibel Island, FL, USA*. Ed. by John H. Gennari, Bruce W. Porter, and Yolanda Gil. ACM, 2003, pp. 70–77.

[162] Andreea I Niculescu et al. "Design and evaluation of a conversational agent for the touristic domain". In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*. IEEE. 2014, pp. 1–10.

[163] A. Nie, E. D. Bennett, and N. D. Goodman. "Learning to Explain: Answering Why-Questions via Rephrasing". In: *CoRR* 1906.01243 (2019).

[164] Natasha Noy et al. “Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it’s done”. In: *Queue* 17.2 (2019), pp. 48–75.

[165] Reham A. Osama, Nagwa M. El-Makky, and Marwan Torki. “Question Answering Using Hierarchical Attention on Top of BERT Features”. In: *MRQA@EMNLP*. Association for Computational Linguistics, 2019, pp. 191–195.

[166] Shaha T Al-Otaibi and Mourad Ykhlef. “A survey of job recommender systems”. In: *International Journal of the Physical Sciences* 7.29 (2012), pp. 5127–5142.

[167] Arantxa Otegi et al. “Conversational Question Answering in Low Resource Scenarios: A Dataset and Case Study for Basque”. In: *LREC*. European Language Resources Association, 2020, pp. 436–442.

[168] Endang Wahyu Pamungkas. “Emotionally-aware chatbots: A survey”. In: *arXiv preprint arXiv:1906.09774* (2019).

[169] Kishore Papineni et al. “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[170] Paolo Pastore et al. “A General Aspect-Term-Extraction Model for Multi-Criteria Recommendations (Long paper)”. In: *Joint Workshop Proceedings of the 3rd Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) and the 5th Edition of Recommendation in Complex Environments (ComplexRec) co-located with 15th ACM Conference on Recommender Systems (RecSys 2021), Virtual Event, Amsterdam, The Netherlands, September 25, 2021*. Ed. by Vito Walter Anelli et al. Vol. 2960. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[171] Rivindu Perera and Parma Nand. “Interaction History Based Answer Formulation for Question Answering”. In: *KESW*. Vol. 468. Communications in Computer and Information Science. Springer, 2014, pp. 128–139.

[172] Olga Perski et al. “Does the addition of a supportive chatbot promote user engagement with a smoking cessation app? An experimental study”. In: *Digital health* 5 (2019), p. 2055207619880676.

[173] Matthew E. Peters et al. “Deep Contextualized Word Representations”. In: *NAACL-HLT*. Association for Computational Linguistics, 2018, pp. 2227–2237.

[174] Matthew E. Peters et al. “Deep contextualized word representations”. In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365.

[175] Maxime Petit et al. “The coordinating role of language in real-time multimodal learning of cooperative tasks”. In: *IEEE Transactions on Autonomous Mental Development* 5.1 (2012), pp. 3–17.

[176] Fabio Petroni et al. “Language Models as Knowledge Bases?” In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 2463–2473.

[177] V. Petukhova et al. “Understanding Questions and Extracting Answers: Interactive Quiz Game Application Design”. In: *LTC*. Lecture Notes in Computer Science. Springer, 2015, pp. 246–261.

[178] Antoine Piau et al. “A smartphone Chatbot application to optimize monitoring of older patients with cancer”. In: *International journal of medical informatics* 128 (2019), pp. 18–23.

[179] Marco Polignano et al. “AlBERTo: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets”. In: *Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019*. Ed. by Raffaella Bernardi, Roberto Navigli, and Giovanni Semeraro. Vol. 2481. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[180] Maria Pontiki et al. “SemEval-2014 Task 4: Aspect Based Sentiment Analysis”. In: *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*. Ed. by Preslav Nakov and Torsten Zesch. The Association for Computer Linguistics, 2014, pp. 27–35.

[181] Maria Pontiki et al. "SemEval-2016 Task 5: Aspect Based Sentiment Analysis". In: *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. Ed. by Steven Bethard et al. The Association for Computer Linguistics, 2016, pp. 19–30.

[182] Soujanya Poria, Erik Cambria, and Alexander F. Gelbukh. "Aspect extraction for opinion mining with a deep convolutional neural network". In: *Knowl. Based Syst.* 108 (2016), pp. 42–49.

[183] Marco Pota et al. "An Effective BERT-Based Pipeline for Twitter Sentiment Analysis: A Case Study in Italian". In: *Sensors* 21.1 (2021), p. 133.

[184] Akshit Pradhan et al. "Enhancing Interaction with Social Networking Sites for Visually Impaired People by Using Textual and Visual Question Answering". In: *CICBA (2)*. Vol. 1031. Communications in Computer and Information Science. Springer, 2018, pp. 3–14.

[185] Zihao Qi et al. "Incorporate User Representation for Personal Question Answer Selection Using Siamese Network". In: *ICASSP*. IEEE, 2019, pp. 7540–7544.

[186] Chen Qu et al. "Attentive History Selection for Conversational Question Answering". In: *CIKM*. ACM, 2019, pp. 1391–1400.

[187] Chen Qu et al. "BERT with History Answer Embedding for Conversational Question Answering". In: *SIGIR*. ACM, 2019, pp. 1133–1136.

[188] Chen Qu et al. "Open-Retrieval Conversational Question Answering". In: *SIGIR*. ACM, 2020, pp. 539–548.

[189] Alec Radford et al. "Improving language understanding by generative pre-training". In: *OpenAI blog* (2018), p. 12.

[190] Filip Radlinski and Nick Craswell. "A theoretical framework for conversational search". In: *Proceedings of the 2017 conference on conference human information interaction and retrieval*. 2017, pp. 117–126.

[191] Sheizaf Rafaeli, Fay Sudweeks, et al. “Interactivity on the Nets”. In: *Network and netplay: Virtual groups on the Internet* (1998), pp. 173–189.

[192] Dimitrios Rafailidis. “The Technological Gap Between Virtual Assistants and Recommendation Systems”. In: *arXiv:1901.00431* (2018).

[193] Rachael Rafter and Barry Smyth. “Passive profiling from server logs in an online recruitment environment”. In: *Workshop on Intelligent Techniques for Web Personalization at the the 17th International Joint Conference on Artificial Intelligence, Seattle, Washington, USA, August, 2001*. 2001.

[194] Pranav Rajpurkar, Robin Jia, and Percy Liang. “Know What You Don’t Know: Unanswerable Questions for SQuAD”. In: *ACL (2)*. Association for Computational Linguistics, 2018, pp. 784–789.

[195] Pranav Rajpurkar et al. “SQuAD: 100, 000+ Questions for Machine Comprehension of Text”. In: *EMNLP*. The Association for Computational Linguistics, 2016, pp. 2383–2392.

[196] Pranav Rajpurkar et al. “Squad: 100,000+ questions for machine comprehension of text”. In: *arXiv preprint arXiv:1606.05250* (2016).

[197] Siva Reddy, Danqi Chen, and Christopher D. Manning. “CoQA: A Conversational Question Answering Challenge”. In: *Trans. Assoc. Comput. Linguistics* 7 (2019), pp. 249–266.

[198] Hyekyun Rhee et al. “Mobile phone-based asthma self-management aid for adolescents (mASMAA): a feasibility study”. In: *Patient preference and adherence* 8 (2014), p. 63.

[199] Heather Riley and Mohan Sridharan. “Integrating Non-monotonic Logical Reasoning and Inductive Learning With Deep Learning for Explainable Visual Question Answering”. In: *Frontiers Robotics AI* 6 (2019), p. 125.

[200] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. “A Primer in BERTology: What We Know About How BERT Works”. In: *Trans. Assoc. Comput. Linguistics* 8 (2020), pp. 842–866.

[201] Andreas Rücklé and Iryna Gurevych. "End-to-End Non-Factoid Question Answering with an Interactive Visualization of Neural Attention Weights". In: *ACL (System Demonstrations)*. Association for Computational Linguistics, 2017, pp. 19–24.

[202] Amrita Saha et al. "Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph". In: *arXiv:1801.10314* (2018).

[203] Wataru Sakata et al. "FAQ Retrieval using Query-Question Similarity and BERT-Based Query-Answer Relevance". In: *SIGIR*. ACM, 2019, pp. 1113–1116.

[204] Malte Schwarzer et al. "An Interactive e-Government Question Answering System". In: *LWDA*. Vol. 1670. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 74–82.

[205] Bhaskar Sen, Nikhil Gopal, and Xinwei Xue. "Support-BERT: Predicting Quality of Question-Answer Pairs in MSDN using Deep Bidirectional Transformer". In: *CoRR* abs/2005.08294 (2020).

[206] Sofia Serrano and Noah A. Smith. "Is Attention Interpretable?" In: *ACL (1)*. Association for Computational Linguistics, 2019, pp. 2931–2951.

[207] Lifeng Shang, Zhengdong Lu, and Hang Li. "Neural responding machine for short-text conversation". In: *arXiv:1503.02364* (2015).

[208] Huan Shao et al. "Intra-Modality Feature Interaction Using Self-attention for Visual Question Answering". In: *ICONIP (5)*. Vol. 1143. Communications in Computer and Information Science. Springer, 2019, pp. 215–222.

[209] Tao Shen et al. "Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base". In: *EMNLP-IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 2442–2451.

[210] Lei Shi et al. "Multi-Layer Content Interaction Through Quaternion Product for Visual Question Answering". In: *ICASSP*. IEEE, 2020, pp. 4412–4416.

[211] Andrew Shin, Yoshitaka Ushiku, and Tatsuya Harada. "Customized Image Narrative Generation via Interactive Visual Question Generation and Answering". In: *CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 8925–8933.

[212] W. Siblini et al. "Multilingual Question Answering from Formatted Text applied to Conversational Agents". In: *CoRR* 1910.04659 (2019).

[213] Marie A Sillice et al. "Using relational agents to promote exercise and sun protection: assessment of participants' experiences with two interventions". In: *Journal of medical Internet research* 20.2 (2018), e7640.

[214] Brent Smith and Greg Linden. "Two decades of recommender systems at amazon. com". In: *Ieee internet computing* 21.3 (2017), pp. 12–18.

[215] Richard Socher et al. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[216] Minchae Song, Hyunjung Park, and Kyung-shik Shin. "Attention-based long short-term memory network using sentiment lexicon embedding for aspect-level sentiment analysis in Korean". In: *Inf. Process. Manag.* 56.3 (2019), pp. 637–653.

[217] Alessandro Sordoni et al. "A neural network approach to context-sensitive generation of conversational responses". In: *arXiv preprint arXiv:1506.06714* (2015).

[218] Daniil Sorokin and Iryna Gurevych. "Interactive Instance-based Evaluation of Knowledge Base Question Answering". In: *EMNLP (Demonstration)*. Association for Computational Linguistics, 2018, pp. 114–119.

[219] Harald Steck. "Calibrated recommendations". In: *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. 2018, pp. 154–162.

[220] Lixin Su et al. "An Adaptive Framework for Conversational Question Answering". In: *AAAI*. AAAI Press, 2019, pp. 10041–10042.

[221] Yu Su et al. "Natural Language Interfaces with Fine-Grained User Interaction: A Case Study on Web APIs". In: *SIGIR*. ACM, 2018, pp. 855–864.

[222] Sethuramalingam Subramaniam et al. "Cobots-a cognitive multi-bot conversational framework for technical support". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 597–604.

[223] H. Sugiyama, T. Meguro, and R. Higashinaka. "Evaluation of Question Answering System About Conversational Agent's Personality". In: *IWSDS*. Lecture Notes in Electrical Engineering. Springer, 2016, pp. 183–194.

[224] Wilson L Taylor. ""Cloze procedure": A new tool for measuring readability". In: *Journalism quarterly* 30.4 (1953), pp. 415–433.

[225] Ian Tenney et al. "What do you learn from context? Probing for sentence structure in contextualized word representations". In: *ICLR (Poster)*. OpenReview.net, 2019.

[226] Robert L Thorndike. "Who belongs in the family?" In: *Psychometrika* 18.4 (1953), pp. 267–276.

[227] Kristina Toutanova and Danqi Chen. "Observed versus latent features for knowledge base and text inference". In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality* (2015), pp. 57–66.

[228] Théo Trouillon et al. "Complex Embeddings for Simple Link Prediction". In: *ICML*. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2071–2080.

[229] Aditya Nrusimha Vaidyam et al. "Chatbots and conversational agents in mental health: a review of the psychiatric landscape". In: *The Canadian Journal of Psychiatry* 64.7 (2019), pp. 456–464.

[230] Svitlana Vakulenko et al. “Question Rewriting for Conversational Question Answering”. In: *WSDM*. ACM, 2021, pp. 355–363.

[231] Danny Suarez Vargas, Lucas Rafael Costella Pessutto, and Viviane Pereira Moreira. “Simple Unsupervised Similarity-Based Aspect Extraction”. In: *CoRR* abs/2008.10820 (2020). arXiv: 2008.10820.

[232] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[233] Anusha Vegesna, Pranjal Jain, and Dhruv Porwal. “Ontology based chatbot (for e-commerce website)”. In: *International Journal of Computer Applications* 179.14 (2018), pp. 51–55.

[234] Jesse Vig. “A Multiscale Visualization of Attention in the Transformer Model”. In: *ACL (3)*. Association for Computational Linguistics, 2019, pp. 37–42.

[235] Oriol Vinyals and Quoc Le. “A neural conversational model”. In: *arXiv preprint arXiv:1506.05869* (2015).

[236] Mattias Wahde and Marco Virgolin. “Conversational Agents: Theory and Applications”. In: *CoRR* abs/2202.03164 (2022). arXiv: 2202.03164. URL: https://arxiv.org/abs/2202.03164.

[237] Ulli Waltinger, Alexa Breuing, and Ipke Wachsmuth. “Connecting Question Answering and Conversational Agents - Contextualizing German Questions for Interactive Question Answering Systems”. In: *Künstliche Intell.* 26.4 (2012), pp. 381–390.

[238] A. Wang et al. “GLUE: A multi-task benchmark and analysis platform for natural language understanding”. In: *arXiv:1804.07461* (2018).

[239] Alex Wang et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *BlackboxNLP*. Association for Computational Linguistics, 2018, pp. 353–355.

[240] Quan Wang et al. “Knowledge Graph Embedding: A Survey of Approaches and Applications”. In: *IEEE Trans. Knowl. Data Eng.* 29.12 (2017), pp. 2724–2743.

[241] Yequan Wang et al. “Attention-based LSTM for Aspect-level Sentiment Classification”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. Ed. by Jian Su, Xavier Carreras, and Kevin Duh. The Association for Computational Linguistics, 2016, pp. 606–615.

[242] Z. Wang et al. “Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 5877–5881.

[243] Zhen Wang et al. “Knowledge Graph Embedding by Translating on Hyperplanes”. In: *AAAI*. AAAI Press, 2014, pp. 1112–1119.

[244] Pontus Wärnestl. “User evaluation of a conversational recommender system”. In: *Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*. 2005, pp. 32–39.

[245] Philip Weber and Thomas Ludwig. “(Non-) Interacting with conversational agents: perceptions and motivations of using chatbots and voice assistants”. In: *Proceedings of the Conference on Mensch und Computer*. 2020, pp. 321–331.

[246] Sarah Wiegreffe and Yuval Pinter. “Attention is not not Explanation”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 11–20.

[247] Jason Williams, Antoine Raux, and Matthew Henderson. “The Dialog State Tracking Challenge Series: A Review”. In: *Dialogue & Discourse* 7.3 (Apr. 2016), pp. 4–33. ISSN: 2152-9620.

[248] Rainer Winkler and Matthias Söllner. “Unleashing the potential of chatbots in education: A state-of-the-art analysis”. In: *Academy of Management Annual Meeting (AOM)*. 2018.

[249] Wilson Wong, John Thangarajah, and Lin Padgham. “Health conversational system based on contextual matching of community-driven question-answer pairs”. In: *CIKM*. ACM, 2011, pp. 2577–2580.

[250] Wilson Wong et al. “Mixed-initiative conversational system using question-answer pairs mined from the web”. In: *CIKM*. ACM, 2012, pp. 2707–2709.

[251] Wilson Wong et al. “Strategies for Mixed-Initiative Conversation Management using Question-Answer Pairs”. In: *COLING*. Indian Institute of Technology Bombay, 2012, pp. 2821–2834.

[252] Chuhan Wu et al. “A hybrid unsupervised method for aspect term and opinion target extraction”. In: *Knowl. Based Syst.* (2018), pp. 66–73.

[253] Fei Wu et al. “Temporal Interaction and Causal Influence in Community Based Question Answering”. In: *IEEE Trans. Knowl. Data Eng.* 29.10 (2017), pp. 2304–2317.

[254] Jinmeng Wu et al. “Building interactive sentence-aware representation based on generative language model for community question answering”. In: *Neurocomputing* 389 (2020), pp. 93–107.

[255] Yonghui Wu et al. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: *CoRR* abs/1609.08144 (2016).

[256] Zhiyong Wu et al. “PERQ: Predicting, Explaining, and Rectifying Failed Questions in KB-QA Systems”. In: *WSDM*. ACM, 2020, pp. 663–671.

[257] Zhiyong Wu et al. “Perturbed Masking: Parameter-free Probing for Analyzing and Interpreting BERT”. In: *ACL*. Association for Computational Linguistics, 2020, pp. 4166–4176.

[258] Patrick Xia, Shijie Wu, and Benjamin Van Durme. “Which *BERT? A Survey Organizing Contextualized Encoders”. In: *EMNLP (1)*. Association for Computational Linguistics, 2020, pp. 7516–7533.

[259] Zhipeng Xie. “Enhancing Document-Based Question Answering via Interaction Between Question Words and POS Tags”. In: *NLPCC*. Lecture Notes in Computer Science. Springer, 2017, pp. 136–147.

[260] Kun Xiong et al. “Neural Contextual Conversation Learning with Labeled Question-Answering Pairs”. In: *CoRR* abs/1607.05809 (2016).

[261] Jingjing Xu et al. “Asking Clarification Questions in Knowledge-Based Question Answering”. In: *EMNLP-IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 1618–1629.

[262] Bishan Yang et al. “Embedding Entities and Relations for Learning and Inference in Knowledge Bases”. In: *ICLR (Poster)*. 2015.

[263] Wei Yang et al. “Data Augmentation for BERT Fine-Tuning in Open-Domain Question Answering”. In: *CoRR* abs/1904.06652 (2019).

[264] Wei Yang et al. “End-to-End Open-Domain Question Answering with BERTserini”. In: *NAACL-HLT (Demonstrations)*. Association for Computational Linguistics, 2019, pp. 72–77.

[265] Zekun Yang et al. “BERT Representations for Video Question Answering”. In: *WACV*. IEEE, 2020, pp. 1545–1554.

[266] Liang Yao, Chengsheng Mao, and Yuan Luo. “KG-BERT: BERT for Knowledge Graph Completion”. In: *CoRR* abs/1909.03193 (2019).

[267] Hai Ye et al. “Dependency-Tree Based Convolutional Neural Networks for Aspect Term Extraction”. In: *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part II*. Ed. by Jinho Kim et al. Lecture Notes in Computer Science. 2017, pp. 350–362.

[268] Xingdi Yuan et al. “Interactive Language Learning by Question Answering”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 2796–2813.

[269] Hongzhi Zhang et al. “An Attention-Based Word-Level Interaction Model: Relation Detection for Knowledge Base Question Answering”. In: *CoRR* abs/1801.09893 (2018).

[270] Sheng Zhang et al. “Multi-Scale Attentive Interaction Networks for Chinese Medical Question Answer Selection”. In: *IEEE Access* 6 (2018), pp. 74061–74071.

[271] Sheng Zhang et al. "Record: Bridging the gap between human and machine commonsense reading comprehension". In: *arXiv preprint arXiv:1810.12885* (2018).

[272] Xinbo Zhang, Lei Zou, and Sen Hu. "An Interactive Mechanism to Improve Question Answering Systems via Feedback". In: *CIKM*. ACM, 2019, pp. 1381–1390.

[273] Yingying Zhang et al. "Multi-modal Knowledge-aware Hierarchical Attention Network for Explainable Medical Question Answering". In: *ACM Multimedia*. ACM, 2019, pp. 1089–1097.

[274] Tiancheng Zhao and Maxine Eskenazi. "Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning". In: *arXiv:1606.02560 [cs]* (June 2016). arXiv: 1606.02560. URL: http://arxiv.org/abs/1606.02560.

[275] Weiguo Zheng et al. "Interactive natural language question answering over knowledge graphs". In: *Inf. Sci.* 481 (2019), pp. 141–159.

[276] Zehai Zhou. "A framework for virtual assistants: An exploratory study". In: *International Journal of Social Science and Business* 1.4 (2016), pp. 49–56.

[277] Zhiheng Zhou et al. "Single turn Chinese emotional conversation generation based on information retrieval and question answering". In: *IALP*. IEEE, 2017, pp. 103–106.

[278] Chenguang Zhu, Michael Zeng, and Xuedong Huang. "SDNet: Contextualized Attention-based Deep Network for Conversational Question Answering". In: *CoRR* abs/1812.03593 (2018).