NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Chatbots: History, uses, classification and response pool generation techniques

Ioannis N. Mitrou

Supervisor: **Panagiotis Stamatopoulos,** Assistant Professor

ATHENS

February 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πράκτορες συζήτησης (chatbots) : Ιστορία, χρήσεις, ταξινόμηση και τεχνικές δημιουργίας λίστας απαντήσεων

Ιωάννης Ν. Μήτρου

Επιβλέπων : **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

Φεβρουάριος 2021

BSc THESIS

Chatbots: History, uses, classification and response pool generation techniques

Ioannis N. Mitrou

S.N.: 1115201400108

SUPERVISOR: **Panagiotis Stamatopoulos,** Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πράκτορες συζήτησης (chatbots) : Ιστορία, χρήσεις, ταξινόμηση και τεχνικές δημιουργίας λίστας απαντήσεων

Ιωάννης Ν. Μήτρου

Α.Μ.: 1115201400108

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

ABSTRACT

Rapidly evolving technology has brought chatbots (or computer conversational agents) back in the limelight. Although chatbots have existed for many decades tracing back to the Turing Test and the ELIZA chatbot, Machine Learning techniques have allowed more complex and more accurate chatbot implementations, resulting in chatbots dominating the customer support field in many businesses and showing up in virtual assistants, but also with a high potential of benefiting education, health and other important facets of modern life.

This thesis will first delve into chatbot applications, their history, as well as their popular uses and fields they should be used in. Subsequently, emphasis will be placed on their classification based on implementation and based on range of topics they specialize in. Then, the methods of evaluating a chatbot will be elaborated upon and a closed-domain retrieval chatbot model will be analyzed.

Finally, in the focus of the thesis, we analyze and present our own techniques for generating a response pool for a more efficient retrieval model and we evaluate the results.

SUBJECT AREA: Artificial Intelligence, Natural Language Processing

KEYWORDS: Chatbot, Conversational agents, task-oriented, open-domain, retrieval, chatbot evaluation, Latent Dirichlet Allocation, clustering

ΠΕΡΙΛΗΨΗ

Η ραγδαία ανάπτυξη της τεχνολογίας έφερε τα chatbots (ή πράκτορες συζήτησης) πάλι στο προσκήνιο. Παρόλο που τα chatbots προϋπήρχαν για πολλές δεκαετίες ξεκινώντας από το Turing Test και τον πράκτορα συζήτησης ELIZA, σύγχρονες τεχνικές Μηχανικής Μάθησης έχουν επιτρέψει πιο σύνθετες και ακριβείς υλοποιήσεις με αποτέλεσμα την πολύ συχνή εμφάνιση εφαρμογών chatbots σε εικονικούς βοηθούς και υποστήριξη πελατών επιχειρήσεων, αλλά και σε σημαντικούς τομείς της σύγχρονης ζωής, όπως στην υγεία και εκπαίδευση.

Η εργασία θα εξετάσει πρώτα τις εφαρμογές chatbot, το ιστορικό τους, καθώς και τις δημοφιλείς χρήσεις τους και τα πεδία στα οποία μπορούν να χρησιμοποιηθούν. Στη συνέχεια, θα δοθεί έμφαση στην ταξινόμησή τους με βάση την υλοποίηση και με βάση το εύρος των θεμάτων που ειδικεύονται. Έπειτα, θα εξεταστούν οι μέθοδοι αξιολόγησης ενός chatbot και θα αναλυθεί ένα κλειστό μοντέλο ανάκτησης chatbot.

Τέλος, στο επίκεντρο της εργασίας, αναλύουμε και παρουσιάζουμε τις δικές μας τεχνικές για τη δημιουργία μιας λίστας απαντήσεων για ένα πιο αποτελεσματικό μοντέλο ανάκτησης και αξιολογούμε τα αποτελέσματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη, Επεξεργασία Φυσικής Γλώσσας

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: πράκτορας συζήτησης, chatbot, προσανατολισμένο στην εργασία, ανοιχτού τομέα, μοντέλο ανάκτησης, αξιολόγηση chatbot, Latent Dirichlet Allocation, συσταδοποίηση

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον κ. Σταματόπουλο για τη συνεργασία του, τη βοήθειά του και τη δημιουργική ελευθερία που μου έδωσε στην εργασία.

CONTENTS

LIST OF FIGURES

1 HISTORY OF CHATBOTS AND THEIR USAGE

1.1 Introduction

Before we get any further, we should first define what a **chatbot** is. Chatbots are computer programs that act as conversational agents that interact with the user in natural language and are used more and more in applications, websites and other aspects of technology mostly to provide customer support but also for entertainment and pedagogic/health reasons. With the rise of Machine Learning, chatbots have become the new trend in Artificial Intelligence and Machine Learning. It was estimated in a survey by Business Insider that by 2020-2021, approx. 80% of all businesses will support some kind of a conversational agent [1].

In addition to the growing popularity and demand of conversational apps and their use for marketing and customer support, another reason for the importance of the analysis and understanding of chatbots, is because it constitutes a prime example of the progress and the essence of Artificial Intelligence, as it is the continuation of the Turing test, which will be elaborated on below.

The rapid increase in the utilization of chatbots is mainly due to the improved implementation methods that make use of Machine Learning techniques and models and the flexibility of neural networks. The ability to learn from previous conversations is the “key” that unlocks the possibilities for free-form conversations with a machine. Therefore, most modern chatbots are either retrieval or generative and use such techniques. However, chatbots are not a new idea. The humble beginnings of chatbots are defined by rule-based algorithms that are much simpler to implement.

1.2 Historical Context

1.2.1 Turing Test and ELIZA

The first recorded chatbot may not have existed until 1966, but the foundations of chatbot applications, as well as Artificial Intelligence as a whole, lay in the work of Alan M. Turing. In a paper centered around the question of whether machines can think, Alan Turing introduced in 1950 the now famous **Turing Test** or as he called it, the imitation Game [2]. According to Turing himself, the question of whether machines can think is too meaningless by itself, but the imitation game poses a much more interesting question, which is whether the computer can do well in the following game. In the Turing Test or the imitation game, a person called “the Interrogator”, asks questions and receives answers from a computer and from a human, both being behind closed curtains, making it so the interrogator cannot distinguish the sender. The interrogator must then decide who is the computer and who is a person, based solely on the answers. Interestingly, Alan Turing predicted in 1950 that by the year 2000 technological progress would produce computing machines with a storage capacity of 10^9 bits, and that with such machinery, a computer program would be able to fool the average questioner for 5 minutes about 70% of the time [2].

His exact words were “I believe that in about fifty years’ time it will be possible to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 percent chance of making the right identification after five minutes of questioning. We believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.” His prediction was wrong, of course but it still motivated many researches and developers to strive to defeat the Turing Test in the annual Loebner prize competition, a competition that was structured around the Turing test [3].

Although it is not the main goal of every chatbot, it is highly desirable for them to portray the illusion of a human being (even for a Q&A chatbot for example).

In response to the Turing Test, MIT professor Joseph Weizenbaum implemented in 1966 the first chatbot in history called ELIZA. The chatbot acted in a similar way to a Rogerian (client-centered) therapist rephrasing the client’s answers as followup questions. It was running on “DOCTOR” script and according to Weizenbaum himself, was specifically aimed at simulating a parody of a therapy session by act, mainly because of the simple nature of the techniques employed by ELIZA to portray emotional interaction [4, 5].

The Chatbot was implemented based on rules and matching specific patterns. Weizenbaum developed the chatbot using SLIP (Symmetric List Processor), a language created by the professor in order for it to be used as an extension to FORTRAN for a better and easier manipulation/modification of doubly linked lists. Simple pattern recognition was used and key words were substituted into standardized questions or comments. Despite the simplicity of the chatbot model, the relatively bad performance and the warning of Weizenbaum himself, it is curious that many users got emotionally attached to ELIZA, some even going as far to ask Dr. Weizenbaum to leave the room to grant them privacy. The attribution of human-like characteristics to computer and machines has been known as the ELIZA effect since 1996 [4]. A study by Cristea et al. [4] researching the Eliza Effect in psychotherapy imitated the

Turing test by asking therapists (novice and experts) to rate psychologists (one being real and the other being the ELIZA chatbot without them knowing). While ELIZA was unsurprisingly rated almost unanimously lower, it is astounding that not one out of 138 therapists mentioned that the interaction with ELIZA chatbot displayed unusual features or that it was an unnatural therapist-patient interaction. Although contentious by many, it is regarded by many that ELIZA was the first chatbot to pass the Turing test.

In conclusion, it is apparent that people have a tendency to attribute human characteristics to machines that use natural language and despite the simplicity of the ELIZA chatbot, it laid the early foundations of the perennial search for the perfect computer conversational agent.

A short conversation with a bot based on the ELIZA chatbot using the site http://psych.fullerton.edu/mbirnbaum/psych101/Eliza.htm is presented in Figure 1-1:

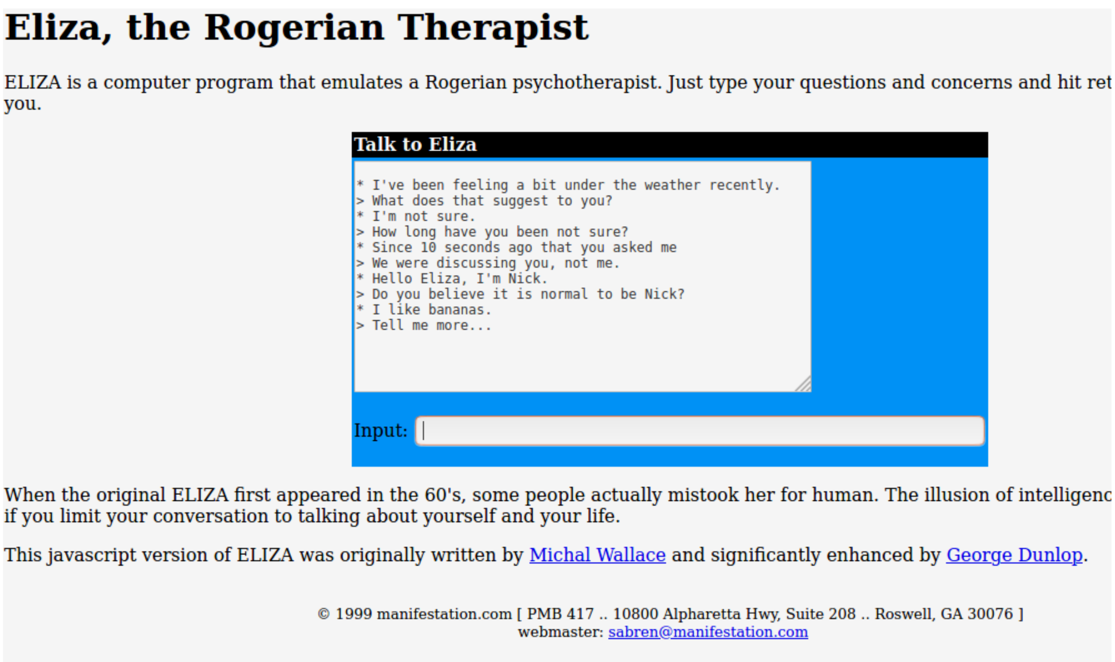

Figure 1-1: A conversation with an ELIZA-based chatbot

1.2.2 Parry

Parry is another well-known chatbot, which was introduced by Kenneth Mark Colby, a psychiatrist and computer scientist, at Stanford's Psychiatry Department in 1972 [6] and it was basically the continuation of the ELIZA research. Colby's approach was the reverse of Weizenbaum's. The program did not behave as a doctor, but as a paranoid schizophrenic patient [6]. It served as a powerful and didactic tool for young psychiatrists and as a functional model of Colby's theoretical pattern of paranoia functioning as a defective processing of signs in a patient's mind.

Our own conversation with the Parry chatbot showcases this tendency, although it is not always apparent if the miscommunication and incoherence are attributed to the schizophrenic nature of the persona the chatbot is trying to imitate, or simply the chatbots inability to understand the question.

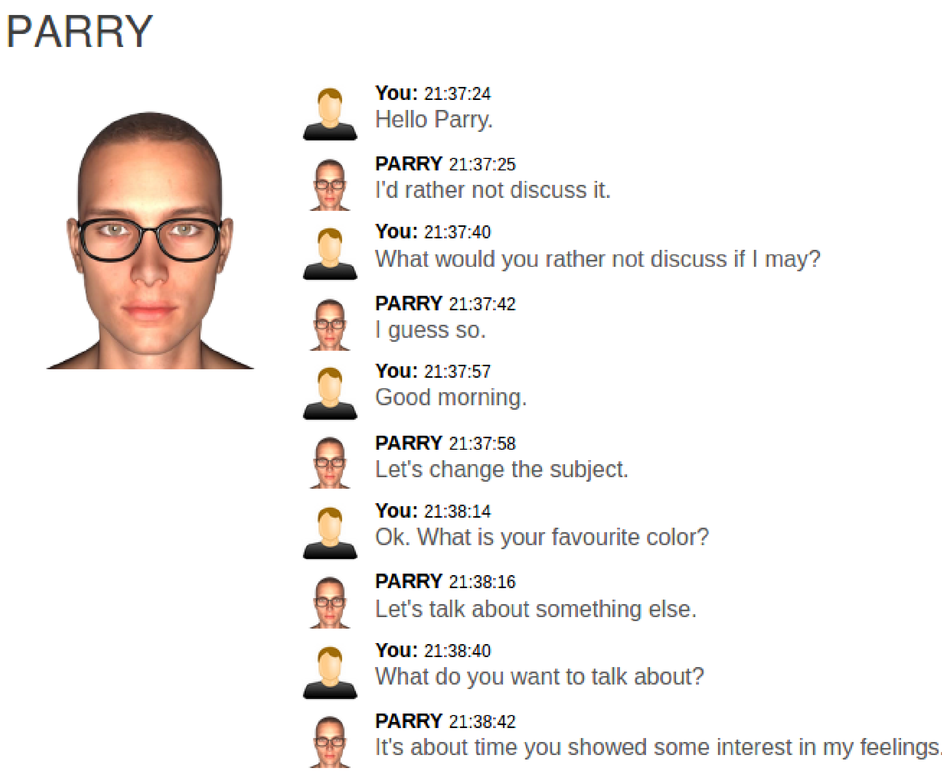

Figure 1-2: A conversation with Parry using botlibre

Since Parry's and Eliza's approach mirror each other but try to achieve the same goal, it was only expected that the 'patient' (Parry) would converse with the 'doctor' (Eliza). At a computer conference in 1972 the two chatbots were pitted against each other and the resulting conversation is shown in Figure 1-3 [6]. The conversation is of great interest as it shows in full effect the Rogerian nature of the Eliza chatbot and the "all over the place" answers and questions of the Parry chatbot, resulting in a comical conversation.

parry: People get on my nerves sometimes.
eliza: I am not sure I understand you fully.
parry: You should pay more attention.
eliza: Suppose you should pay more attention?
parry: You're entitled to your opinion.
eliza: What makes you think I'm entitled to my opinion?
parry: It bothers me just to be around people.
eliza: Please go on.
[Parry continues with some paranoid ramblings about horseracing, bookies, and the Mafia]
parry: I know the mob controls the big rackets.
eliza: Who are you really talking about?
parry: Have you ever really thought about the Mafia?
eliza: Suppose you ever really thought about the Mafia?
parry: Let's talk about something else.
eliza: I am not sure I understand you fully.
parry: You keep asking me the same things, why not change the topic?
eliza: In what way?“. [6]

Figure 1-3: Conversation between ELIZA and Parry

1.2.3 Dr. Sbaitso

In 1991, another milestone in chatbot history came with Dr. Sbaitso. Dr. Sbaitso’s importance lies in the fact that it was the first chatbot to use sound to communicate. It used a Sound Blaster card created by Creative Labs (as known in the US) and it was able to synthesize speech [6]. Nowadays, with Apple’s Siri and Amazon’s Alexa, chatbots that communicate through sound might be trivial, but in 1991 it was revolutionary. The Sound Blaster cards were used in IBM PC compatible computers (a term no longer used) and gave way to Windows 95 that then used audio electronics.

After the chatbot introduces itself, it asks for the user’s name and then proceeds to greet the user with it as a first step of impressing you. Much like ELIZA, Dr. Sbaitso tends to repeat your sentences a lot and in our own testing, tends to break the immersion and the illusion of conscience very quickly but it was still impressive for its age. In order for the chatbot to run in modern OS’s, an emulator like DOSbox that emulates x86 intel architectures is needed. Sites like ClassicReload.com incorporate that emulator so no extra preparation is needed and Dr. Sbaitso can be instantaneously fired up.

The site ClassicReload.com was used to take Dr. Sbaitso for a “test drive” with disappointing results by today’s standards. Having said that, every reply was communicated through sound (albeit synthesized and artificial), something the screenshot does not do justice to.

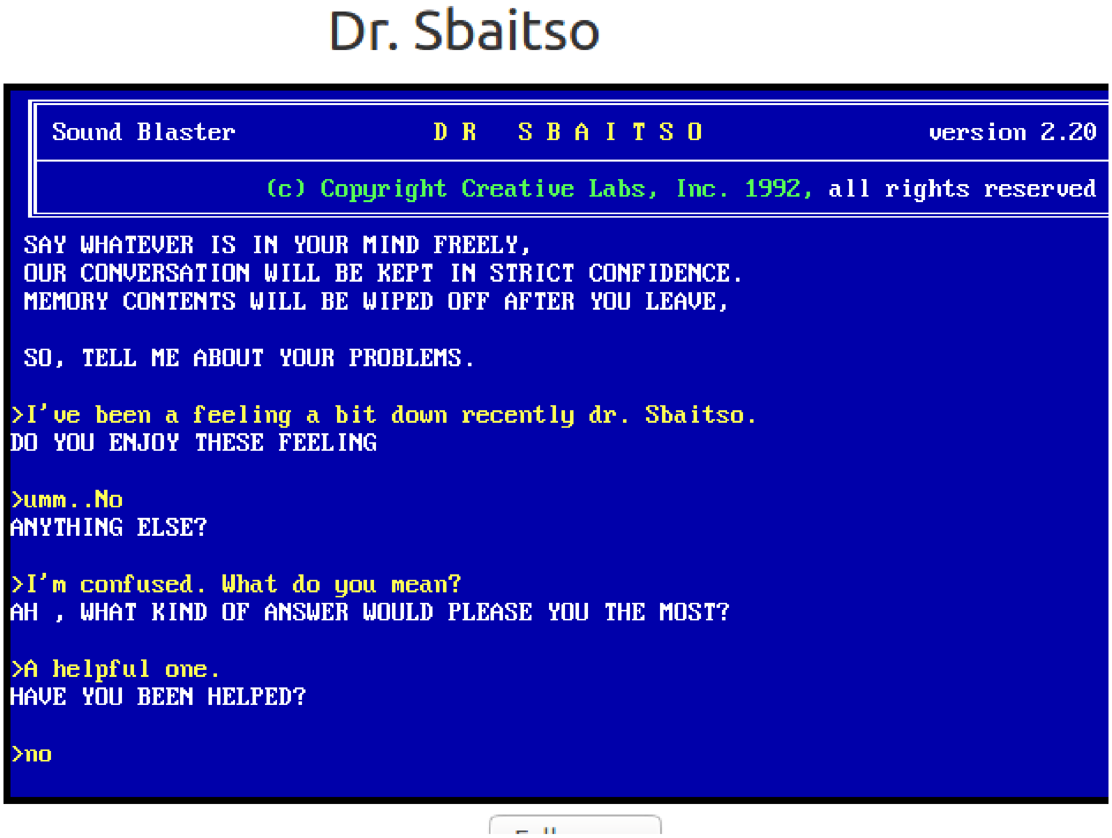

Figure 1-4: A conversation with Dr. Sbaitso using ClassicReload

1.2.4 ALICE

ALICE is the Artificial Linguistic Internet Computer Entity, originated by Wallace in 1995 and arguably the most successful chatbot of its time [7]. It has won the Loebner prize, an annual Artificial Intelligence competition (which has changed these last few years) and although it did not pass the Turing test in the competition (no chatbot did in the early 2000's), it has been judged the most human-like chatbot in 2000, 2001 and 2004.

Alice stores knowledge about English conversation patterns in AIML files. It is probably the strongest of the older rule-based chatbots. It uses the most refined techniques of rule-based algorithms that are explained in detail in rule-based chatbots below.

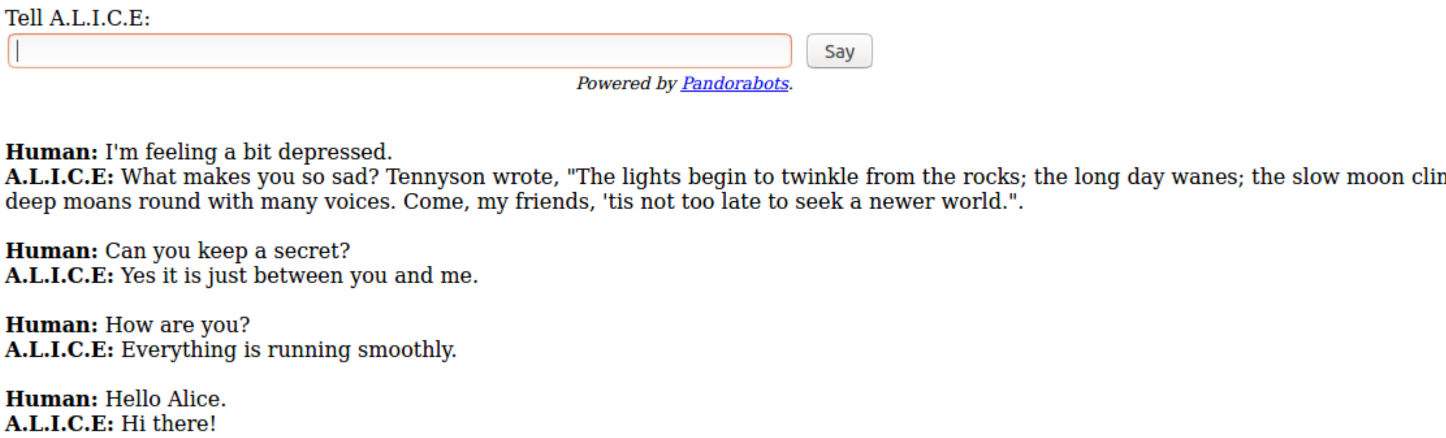

Figure 1-5: A conversation with ALICE using pandorabots

2 CURRENT USES OF CHATBOTS AND THEIR CLASSIFICATION

2.1 State-of-the-Art

Chatbots have become the new trend in Artificial Intelligence and Machine Learning (Google introducing the sequence to sequence model, which will be explained in 2.2.2.2, helped the upcoming boom of popularity). It was estimated by Business Insider in 2016 that by 2020-2021, approx. 80% of all businesses will support some kind of a conversational agent [0], while Gartner stated in 2016 that by 2020 the average person will have more conversations with bots than with their spouse.

A field that has been growing since 1966, is now at its peak with big corporations like Amazon, Microsoft, Google and Apple joining and contributing.

Virtual assistants :

The biggest usage of chatbots today is definitely the usage of virtual assistants, the most popular of which are **Alexa** by Amazon and **Siri** by Apple. According to Siri statistics, there were 500 million devices that used Siri in 2018 and 62% of iPhone users use Siri in their car [8], while Alexa devices that have been sold number to 100 million. Both are chatbots that can use voice interaction and can be viewed as multitools providing users with a multitude of services and applications like music playback, setting alarms and providing weather, traffic and other real-time information. This incredible surge of popularity for the virtual assistants has also helped the general popularity of chatbots and has helped other projects being funded.

2.1.1 Customer support and Financial benefits

Chatbots are most commonly used for customer services, due to the help that they can provide in managing large amounts of data [9], but also due to their ability to handle hundreds or thousands of concurrent users, something that would have required immense resources.

Chatbot implementation not only saves customers service cost by replacing nearly all human assistants, but also increases user satisfaction [9]. Juniper research reported that by 2022, chatbot implementation will save companies worldwide about eight billion dollars annually. The famous hotel chain Marriott International has successfully assisted 44 percent of all registered Facebook users by using a chatbot since 2017 [9].

2.1.2 Artificial Intelligence Competitions

We can clearly see the effect of chatbots in sites, virtual assistants and customer support, but another good indicator for the state of the art of chatbots are the Artificial Intelligence competitions. Loebner prize competition was the most popular one. It started in 1991 [2] with a steadily growing prize and interestingly there are two prizes that have never been awarded. 25.000$ for a program that judges cannot tell apart from a real human and 100.000$ for a program that meets the same standards but also includes deciphering and understanding text, visual and auditory input [2]. Alice chatbot used to be the leading chatbot, but recently Mitsuku has been claiming the top spot for 4 consecutive years.

A different and newer competition is the Amazon Alexa Prize, a 2.5 million dollar university competition that has 16 university teams pitied against each other to make the best socialbot [10]. The competition was announced on Sept. 26 2016 with the purpose of advancing open domain and conversational chatbots [10]. The teams were provided with the Alexa Skills kit to develop the chatbots and the evaluation was different than the Loebner prize competition. The goal was no longer to pass the Turing test, since users already knew that they were conversing with a chatbot. Users rated the chatbot (via the Alexa-let's chat), based on how intelligent and coherent the conversation was.

2.1.3 Entertainment and the Mitsuku Chatbot

The entertainment value of chatbots is also not to be underestimated, as more and more conversational applications emerge that engage with the user in casual conversation like the ones aimed at the Amazon Alexa prize in an enhanced version of the Turing test or the ELIZA experiment. Many of those can be accessed through the web and are free sites or applications. A popular game amongst users of these chatbots has been to make the chatbot admit that is, in fact, a bot. The virtual assistants mentioned in 2.1 also provide entertainment value.

A special mention should be given to Mitsuku, the current Loebner prize winner and one of the most "intelligent" chatbots in the world. It is widely considered the best chatbot for casual conversation and for good reason. A snippet of conversation with Mitsuku is presented below.

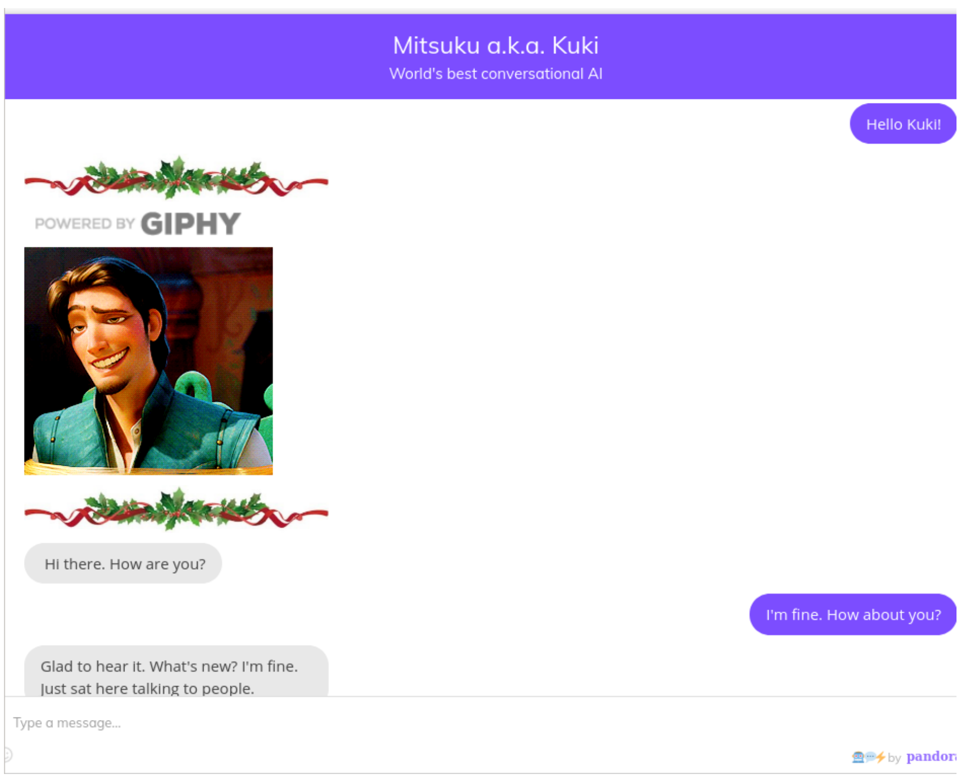

Figure 2-1 A conversation with Mitsuku using pandorabots (Part 1)

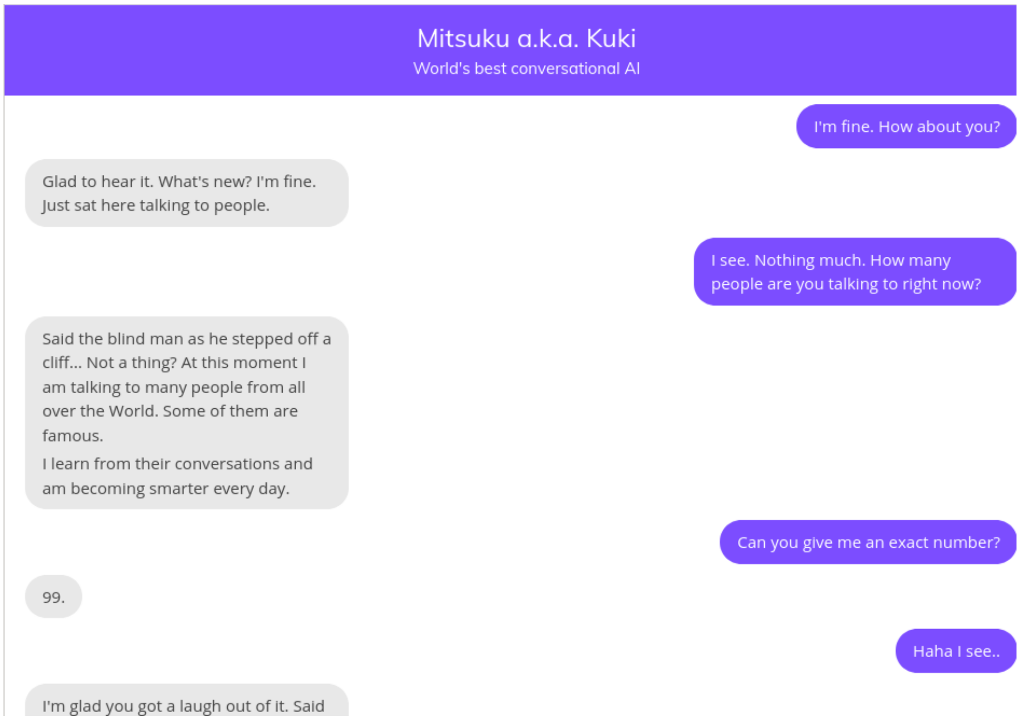

Figure 2-2: A conversation with 'Mitsuku' using pandorabots (Part 2)

Although chatbots are primarily seen in customer support areas, due to their fast improvement and popularity, there is important research being done in other fields such as education and health and these findings are highlighted below.

2.1.4 Health-Mental Health

As we saw in the brief history of chatbots presented above, some of the first major chatbot milestones were chatbots impersonating a psychologist or a psychiatrist. Chatbots and psychiatry/psychology are linked and for good reason: Chatbots might not be a replacement for a psychologist or a human, but they allow the user to speak freely without worrying about human perception or time constraints. As Gale Lucas, a psychologist in USC's Institute for Creative Technologies said, people are very open to feeling connected to things that are not people.

It is evident that conversational agents can be a positive force for mental health and there is no shortage of research backing this [11, 12, 13]. Greer et al. [11] in a study of using chatbots for after cancer treatment on young adults found that most patients rated the experience as helpful and after 4 weeks reported an average reduction in anxiety of 2.58 standardized t-score units (it needs to be said that the sample size was relatively small, numbering 45 young adults).

Ellie chatbot, a chatbot that posed as a psychologist impersonator, much like a modern version of the ELIZA chatbot, had conversations with US war veterans and the results showed that veterans confided more in Ellie than in their official PDHA surveys [12].

Lucas states that “getting people to admit they have symptoms is an important step in helping them realize they are at risk – and getting them treatment” and if chatbots can be the same as or even better than human psychologists at getting people to admit their symptoms, actual psychologists and psychiatrists can spend more time providing them the necessary treatment. Mujeeb et al. [14] proposed Aquabot as an assistant to diagnose achluophobia and autism disorder by asking specific questions.

In conclusion, conversational agents can lend a helping hand to mental doctors and psychologists and could be an even greater benefit in the future. Although clinical psychologists cannot be replaced for the foreseeable future, they can take advantage of this technology to elicit more responses and information from the patients and improve their well-being.

2.1.5 Education

Education is the next logical thing to consider when we think about the advantages of conversational agents. The huge success of chatbots in customer support is already clear, but what if we see education or any important public sector as customer support? Or parts of it at least. There are classes that consist of many students and a single teacher cannot possibly answer to every single question that might pop up. What if the teacher hardcodes some rules into a very simple chatbot that answers some FAQ or provides common learning material and knowledge sources? Chatbots could increase student engagement, especially at the time of writing this thesis that it has been proven how important e-learning is during a global pandemic.

A significant number of studies have already shown successful implementations of chatbots in learning scenarios [15].

Since as early as 2005, Heller et al. tried to show with Freudbot that a chatbot could enhance distance education by having the students chat with a digitalized, emulated Freud [16].

The university of Georgia recently created a chatbot based on IBM’s Watson platform called “Jill Watson”, which was developed to handle forum posts by students enrolled in a computer science course [17]. During the semester, 10.000 questions were submitted. The task of answering every single question by the teacher and their assistants was an impossible one. Therefore, a chatbot was created to answer the simple questions, while the more complex ones would be handled by the professor. The project was a huge success. Students were more engaged and wished they had that option in other courses too.

But Q&A is not the extend of the potential of educational chatbots. Already, there are educational sites that employ evaluation chatbots, evaluating the knowledge of the student, by employing tests that minimize the anxiety that the students have in traditional tests and exams.

[18] studied and summarized in detail all the available literature for the role of the educational chatbot, not just as a bot for answering questions and a separate entity to the professor, but

as an extension to the teaching personnel (with the condition that any chatbot researched is AIML-based). The role of every chatbot was either 'Student Evaluation', 'Q&A', 'Teacher Communication', 'Natural Conversation' or 'Tutor', with the role of Tutor being the most prominent role in the dataset researched. Every role had a set of subfeatures and the most prominent of them were 'Answer general questions', 'Conversational strategies', 'Hold specific topic conversations' and 'Hold general conversation'.

A generative AI chatbot would not be suitable yet (although that might not be the case for the near future) for a course assistant, because the answers need to be perfect. It is preferable for an answer to be irrelevant than incorrect. For now, the best option for chatbot integration into education would probably be a retrieval chatbot that has preset responses and answers accordingly or a AIML-based one like [18] strongly showcases. If the student deems that the answer or the help is too irrelevant, then they can contact the professor directly as a second step, having the chatbot basically act as a intermediary to root out the easiest questions.

Generative and retrieval chatbots will be explained in 2.2.2.2 and 2.2.2.3.

2.2 Chatbot Classification

Since the early days of chatbot development, chatbots have come a long way. Modern chatbots can be categorized in a variety of ways with different criteria, such as how they interact with the user (text or voice) [19], but the most important categorizations to help us understand and analyze them are two [19]:

1. Classification by **range of content and topics** (closed domain and open-domain)
2. Classification by **implementation/design** (rule-based models, retrieval models and generative models)

2.2.1 Closed-domain and open-domain chatbots

Chatbots are easily distinguished by the range of topics they can adapt and answer to. Task-oriented chatbots (or closed-domain chatbots) are built to accommodate a specific function (Q&A chatbots, support chatbots, etc.), while non-task-oriented chatbots (or open-domain chatbots) can simulate an open-ended conversation with a person with no limiting topics [19]. These are mostly used for research and entertainment purposes. To provide an example for each, a Q&A or a telephonic chatbot to book a doctor's appointment is a closed-domain chatbot, while the award winning Mitsuku is an indicative example of an open-domain chatbot.

Another distinction that should be kept in mind is that closed domain chatbots are usually designed to allow for short conversations. The user engages with a closed domain chatbot in order to obtain an information or achieve a task, something that usually happens within a few replies. On the contrary, open-domain chatbots are usually designed to facilitate longer conversations, because their role is often to have casual conversations to entertain the user. Existing task-oriented chatbots vastly outnumber the open-domain chatbots, as they are much easier to implement to achieve a simple goal and most people have used a closed domain chatbot to book a flight, a doctor's appointment or for Q&A for a site or application.

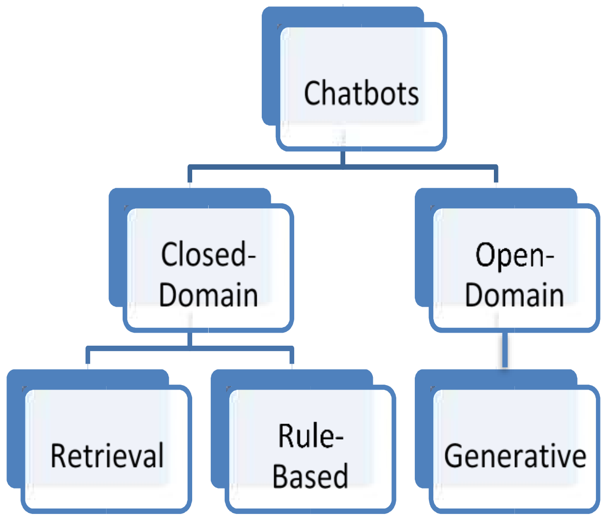

Figure 2-3: Classification of Chatbots by Task Orientation. Last level of the tree indicates the most commonly used design technique. That does not mean for example that retrieval chatbots cannot be open domain or generative chatbots cannot be closed-domain.

Open-domain chatbots lend themselves better to generative design, while closed domain to retrieval, as shown in figure 2-3. That brings us to the next important classification.

2.2.2 Classification by design technique

Depending on the design method and implementation, chatbots are either rule-based, retrieval or generative. Every chatbot ever created falls into one of these categories. Generative and retrieval methods are more recent design techniques that make use of Machine Learning and neural networks, while rule-based implementation was used for earlier chatbots like ELIZA or ALICE. The main concept that differentiates rule-based and neural network-based approaches (generative and retrieval) is the presence of a learning algorithm in the latter case. Retrieval and generative conversational agents both use Machine Learning models but differ in that generative models create answers while retrieval models pick predefined answers depending on the question-context. Rule-based chatbots, on the other hand, have simpler, usually hardcoded rules to carry out simple tasks.

All three classes of chatbots will be elaborated on below.

2.2.2.1 Rule-Based Chatbots

Machine Learning became feasible in 2010s, but before that, most chatbots were using rule-based algorithms, namely algorithms that searched for hardcoded patterns. Like rule-based systems in expert systems in AI, rule-based chatbots function based on rules. To provide an example, if a user asks “What is the price of this <book>”, a rule-based algorithm might match this with the following pattern: “What is the price of” and then just search for the book. Hardcoded and set phrases are matched with the input, using if-else cases.

A more advanced rule-based chatbot is the chatbot named ALICE (A.L.I.C.E ~ Aritificial Linguistic Internet Computer Entity), as mentioned in 1.2 that was developed in AIML (Artificial Intelligence Markup Language) [19]. It uses pattern matching, a technique mostly used for rule-based chatbots. User Input goes through a normalization phase and after its modification, the normalized input is used to create an input path with the following structure:

<normalized input>, “that”, <previous chatbot answer>, “topic”, <current topic>.

This creates a sequence that contains the user input, the previous answer from the chatbot to provide continuity and the current discussion topic to provide context. This is the pre-processing phase.

After preprocessing, the algorithm is trying to match the aforementioned sequence with the largest possible pattern. The larger the sequence matched, the closer and more specific the answer.

This was a more elaborate approach to rule-based algorithms of conversational agents, but there are simpler algorithms with hardcoded questions and responses.

Rule-based chatbots have generally gone out of favor, because of their limited scope and flexibility, but they are not obsolete, as they can be more useful in scenarios requiring speed of training and have limited scope. Their straightforward implementation and ease of use makes them attractive for some businesses and applications. To provide some examples, many customer service and booking applications use rule-based chatbots. Automated phone call receivers for doctor appointments, cinema tickets and restaurant reservations are all made using rule-based algorithms.

2.2.2.2 Generative Chatbots

Generative chatbots, as the name suggests, generate responses by forming sequences of words. Unlike retrieval chatbots (that will be covered in 2.2.2.3), generative chatbots do not receive ready-made responses from a database, but create them from scratch. On the other hand, they require a huge amount of training data, so as not to make too many grammatical mistakes, as unlike retrieval chatbots, they do not have already grammatically correct responses.

The go-to model and probably the reason for the existence of the generative model is now the seq2seq or sequence to sequence model introduced by Google.

The sequence to sequence model consists of two Recurrent Neural Networks that are trained in tandem, an encoder and a decoder [20, 21]. In a seq2seq model, a variable-length concatenated sequence (user input and chat history) is parsed into an RNN neural network (encoder) and produces a fixed-length vector which encaptulates the “meaning” of the original sequence. The output-vector is then seeded as input into another RNN neural network (decoder), in order to produce a variable-length output sequence. (We need two neural networks to have variable length sequences, meaning different lengths between input and output).

The purpose of a generative chatbot is usually to simulate free form discussion, as retrieval chatbots would be limited in open domain casual scenarios.

2.2.2.3 Retrieval Chatbots

Retrieval Chatbots do not generate responses, but score and pick the best response from a set of predefined answers. They simply pick responses from a repository of context-response pairs, and therefore the responses do not contain any unplanned grammatical errors.

The general retrieval model is defined by a search engine that selects the aforementioned set of responses and then a model to rank those responses and select the highest-rated. [20]

Retrieval models may vary in the details, but the most common model architecture consists of a Siamese (RNN or LSTM) neural network that take two kinds of inputs respectively: context and candidate response. Context is defined as the concatenated history of the chatbot-user conversation including the last user input that awaits a reponse from the chatbot and the candidate response is a response from our set that will receive a score. Then the model outputs a score $s(c,r)$, where c = context and r = response, dictating how relevant and appropriate the response is. More details on the model and its training are presented in 3.2.

Let us say the model is now trained. The question then remains, after training how can the model rank every single candidate response to select the best one? The simple answer is, it cannot. But, despite the importance of the method for the selection of a sample pool, little attention has been given to this “whitelist” as Swanson et al. describe it [26], meaning a list of likely responses. Let D be the set of responses stored in a database. The goal is to efficiently find a subset S of D that has a high possibility of containing at least one response that is suitable in this particular scenario. Most papers and researched models focus on finding the best response from a small list of sampled responses, and practical use of the model is not being addressed [26]. The generation of the response pool will be elaborated on in 3.3.

It is also important to distinguish between single-turn response and multiple turn response. Earlier retrieval chatbots used single-turn response , which equates to using only the last utterance as input, in comparison to multipleturn response that use a multi-turn context. Human generated responses are heavily dependent on the previous dialogue segments at different granularities

(words, phrases, sentences, etc.), both semantically and functionally, over multiple turns rather than one turn [21], so that highlights the importance of multi-turn context during the training of the model, making the earlier retrieval models obsolete.

2.2.2.4 Retrieval vs. Generative approach

Unless our goal is a very simple customer support chatbot (e.g. to book a flight or an appointment), generative and retrieval approaches are the only viable approaches in 2020, given the advance in computer architecture and Machine Learning techniques, libraries and models. Thus, it is important to distinguish their advantages and disadvantages in order for us to know when to use each one, as their respective areas sometimes do not overlap.

The obvious distinction that has been mentioned above, is the fact the retrieval chatbots have no grammatical errors or abnormalities, as they always pick from a set of predefined responses. However, the response coverage is restricted, because retrieval models can not handle unseen queries for which predefined responses do not exist [20]. This problem can be alleviated by having a large and diverse set of responses but never eliminated. The chatbot **will** still select the highest- rated response, but it might be highly inaccurate.

Generative models on the other hand are much more capable in open domain conversations, as they create the responses from vectors ("meanings"), but are also much more prone to grammatical and syntactical errors.

Another interesting thing to consider is that retrieval chatbots are much easier to analyze and improve, because of the immediacy and transparency of the results in contrast to the generative approach, the nature of which renders its evaluation much harder. The performance of a retrieval chatbot can be easily evaluated using methods like Recall@k in a case of a specific context and a response that corresponds to that context. If the chatbot selects a different response, then it was incorrect and it continues to the next context-response pair, having a kind of binary evaluation (correct - incorrect). The evaluation of a generative model is not so straightforward (depending on the model used).

2.3 Chatbot Performance and Evaluation

As far as Chatbot evaluation is concerned, it is a multivariant concept that among other factors includes accuracy, user engagement and satisfaction and speed of training as well as response time and is dependent on the implementation technique. There are evaluation metrics that decide how accurate the chatbot is and Recall@k is mentioned later on, but the evaluation of the User Experience is a different thing entirely. Therefore, we measure chatbot performance based on what we want to achieve and what we want to emphasize and it would be counterproductive to decide on a universal performance method. However, every evaluation method is either done by the computer or by humans [22, 35] and that is an important distinction to make.

Different researchers have established different evaluation frameworks, but this information could be distilled in two main and general areas [22]:

1. **Content Evaluation (mostly carried out by computers)**
2. **User Experience Evaluation (carried out by humans)**

2.3.1 Content Evaluation

Content evaluation is more formal than the User Experience Evaluation, evaluating the chatbot's performance and accuracy. Content evaluation can be automatic or Expert [22]. Expert content evaluation which is evaluation done by humans is more rare and more akin to User Experience Evaluation, while automatic content evaluation which is carried through by the computer uses specific techniques-functions, the most dominant of which are **Precision** and **Recall**. Recall@k is explained in detail in part 3. “Precision is referred to as the measurement of how many times the information given to conversation is relevant to the topic of discussion in percentage” [22].

As far as design technique is concerned and as mentioned in 2.2.2.4, retrieval chatbots (and closed domain) are easier to be evaluated, because with a set of predefined answers that correspond to contexts, the disparity is much more obvious in case of error. Generative chatbots (and open domain) are much harder to evaluate and the most reliable method of evaluation is the human one using Likert scores (questionnaires). Therefore, the User Experience is more important in such cases.

2.3.2 User Experience Evaluation

User Experience can be defined as the general feeling and satisfaction the user receives from the use of the application and more specifically, the chatbot. Researchers might inquire the users as to the rating of the experience and to rate their satisfaction on the Likert scale (Likert scores remain the most useful for subjective user experience rating) [22, 24]. It is a very important type of evaluation for any application and even more for a chatbot, which is a very complex system with not a single correct answer [22].

User Experience evaluation could be categorized to session level and turn level [22], with session level being the general experience of the user and their satisfaction during the entire session with the chatbot, while turn-level being the method where users are asked to rate each response. As mentioned previously, questionnaires and ratings on the Likert scale are often used.

While this is the general approach for chatbot user experience evaluation, depending on the situation and context the chatbot is used for, there are multiple different ways to evaluate it. For example, for businesses specifically, we can measure self-service rate (percentage of user sessions that did not end with a contact action after using the chatbot), usage rate per login and satisfaction rate among others [23].

User Experience is a very significant aspect of every application, site or service that engages users in an activity and is perhaps the most important type of evaluation, because no matter how much the technology changes and content evaluation evolves with different techniques, the evaluation of user experience will always be a timeless aspect of evaluating a chatbot application (and any other application or service).

3. STUDY OF A RETRIEVAL CHATBOT – ANALYSIS OF THE MODEL AND METHODS TO IMPROVE ITS EFFICIENCY

Having gone through the history of conversational agents, their applications and their classification, the next part of the thesis will follow a more **practical approach**. We will **not** train or run a chatbot, but we will instead work on Natural Language processing to generate a subset of responses to improve the efficiency of the response database when it is put to use for a retrieval chatbot. As previously mentioned in 2.2.2.3, a retrieval model ranks preset answers and chooses the best one to answer with based on context. In that way, it needs a database of appropriate responses to choose from. Realizing that there is much interest and importance in a list of those prestored answers because of the impossibility of searching through the whole database, we decided to further expand on the research of Swanson et al. [26]. Our methods can be applied on any retrieval model and therefore, we put emphasis on the preprocessing of the dataset for the creation of the response pool and not its direct effect to the chatbot.

We will still analyze a **retrieval chatbot model** for the sake of completeness and continuity. We choose to analyze and explain a predefined model and not train our own, because this particular area is well documented and well researched. The findings of Lowe et al. [25], and their extensive research around the dataset and the retrieval model they constructed in order to test their dataset have left little space for additions or improvements on a student level. On the other hand, the pool of candidate responses is not widely researched and presents significant interest.

Apart from the analysis of the retrieval model, a Dataset consisting of dialogs will also be expanded upon for **two reasons**: First of all, even though we will not train a chatbot model ourselves, it is very important to analyze the dataset used, because the dataset is perhaps the single most important component of the training of a Machine Learning model and for any other researchers/students, we might provide some insight into the availability of freely available datasets for the training of chatbots. Secondly, and even more importantly, because we could not find a database of preset answers for a retrieval model, we will use the responses that Lowe et al. [25] have organized within the **dataset** for our own purposes (the generation of the Response Pool).

Consequently, we will delve into the dataset we will use for the response pool experiments and the hypothetical retrieval model, we will briefly touch on Recurrent and LSTM neural networks (that are needed for the training of the model), analyze the retrieval model and, lastly, we will focus on ways to create the whitelist presenting our own improvements, methods and findings/conclusions.

3.1 The Dataset (for training, testing and generation of the response pool)

To train a chatbot model, a Natural Language Dataset is required and more specifically, a dataset that consists of dialogs (2-person conversations). It is very important that in order to obtain good results, we have to choose a dataset that contains data similar to the area of conversation we want to focus on. And therein lies an important issue. For obvious reasons, for a chatbot to be effective, it needs to be trained with a dataset centered in the particular topic of interest, but making a large enough dataset with dialogue on a specific topic is very hard and requires a large budget, something most students and independent researchers don't have. Existing and freely available chatlogs, like the **Ubuntu corpus**, are goal-oriented domain casual conversations and because of that they are excellent datasets to train a very specific kind of chatbot but not an open-domain one or one with a different specialization. To provide an example, in an excellent thesis by Caros M. [27], a similar problem occurred when they wanted to train a generative chatbot to aid in Alzheimer therapies. They could neither find datasets with reminiscence therapy terms, nor decent datasets with open-domain dialogs.

Consequently, and not unlike many other studies and researches in this field, we will use the Ubuntu Dialogue Corpus [25], which, to the best of our knowledge, is the largest freely available multi-turn-based dialogue corpus. It was constructed using chat logs from Ubuntu-related chat rooms on the Freenode IRC network and although there are many concurrent users in a chatroom, the chat logs were preprocessed using heuristics to produce two-person conversations. As a result, we have a very large training dataset that can also be used for a generative chatbot.

With the help of the very handy work of Rudol Kadlec (or 'rkadlec' on Github [29]), we can download and do some of the preprocessing of the dataset. The generated files after the preprocessing are the **training file,** the **validation file** and the **test file.**

The training file contains the training set which consists of organized triplets: **context**, **response**, **flag**. Their use will be elaborated on in 3.2. The validation file contains the validation set, which provides the means to evaluate the model, in order for the training to stop early so as not to reach overfitting. The validation/evaluation set consists of 11 columns: one utterance and 10 responses, one of them being the correct one and the rest called distractors. The chatbot ranks the responses and if the correct one is in the k-highest ranked responses (hence recall@k), then the chatbot got that "question" right. Finally, the test file evaluates the final model, and is formatted the same way as the validation set.

The structure of the training file (the organized triplets) is presented below via a code section that reads the file into a pandas dataframe and prints the head (the first "batch" of the dataframe).

```
                                             Context  ... Label
0  i think we could import the old comment via rs...  ...     1
1  i 'm not suggest all - onli the one you modifi...  ...     0
2  afternoon all __eou__ not entir relat to warti...  ...     0
3  interest __eou__ grub-instal work with / be ex...  ...     1
4  and becaus python give mark a woodi __eou__ __...  ...     1

[5 rows x 3 columns]
```

Figure 3-1: An image capturing the structure of the training set

As a final note, another alternative to the Ubuntu dataset would be the Cornell – movie dialogs corpus, put together by Cristian Danescu-Niculescu-Mizi and Lilian Lee in Cornell University [28]. This corpus contains 220.579 conversational exchanges between 10.292 pairs of movie characters, but it is vastly outperformed by Ubuntu Corpus which contains 26 million turns of dialog.

3.2 The Retrieval Model

Having adjusted the dataset, a retrieval model will be elaborated on below based on the work of Lowe et al. [25], which will help us better understand the value of the response pool generation techniques for accuracy improvements in later sections. It is pivotal to mention that Lowe et al. use one of many possible retrieval models, but it is one of the most straightforward, hence we choose to showcase it. No matter the specific model, any retrieval model selects a set of responses and selects the highest-rated. Before the model is presented, it would be useful to briefly touch on Recurrent and LSTM Neural Networks, as they are an integral component in the following model.

Recurrent and Long Short Term Memory (LSTM) Neural Networks:

Recurrent Neural Networks can differ from regular feed forward Neural Networks in that they are able to process variable-sequence data and that is why they are paramount for any chatbot model. An RNN has a looping mechanism that allows previous information to be fed to the next step [33]. For example, if an input sequence is “How are you?”, the tokenized “how” is first fed to the network, and for the next step, “are” is fed along with the hidden step from the previous step. Now, the network has information on both the “how” and “are”.

LSTM Neural Networks are recurrent Neural Networks that have gating units to deal with the vanishing gradient problem. In recurrent neural networks, whenever the gradient of the error function is propagated back through a unit of a neural network, it gets scaled by a certain factor, resulting in the gradient blowing up or decaying exponentially over time [32]. This effect is called the vanishing gradient. LSTM can combat it by containing extra information in every cell of the network. This extra information can be stored, read and written just like computer memory. The flow of information inside the cell gets regulated through an input gate, output gate and forget gate.

The model:

The model described in [25] consists of a dual encoder Neural network, basically one siamese network made of two sub-NN that are either Recurrent or LSTM Neural networks, both solutions being viable. In a conversation with a chatbot, the context is defined as the history of the conversation as of this point in time (mentioned in 2.2.2.3) and the utterance is the candidate response that is to be ranked. One Neural Network will be responsible for the context and the other for the utterance. That means that for every question that is addressed to the chatbot, the model concatenates the history of the conversation along with the current question (and thus the current context is constructed) and feeds it together with a candidate response to the network. This will happen for every possible context-utterance pair. The context obviously remains the same but the number of pairs depends on the number of utterances available in the candidate response list. The output of the model is a number between 0 and 1 (a probability).

More specifically, the procedure is as follows: Given a context-utterance pair, their embeddings c, r ∈ R^d are calculated after the pair has been fed as input to the Siamese neural network. The vectors of the final hidden state c, r encapsulate the summary or meaning of the context and response. Using the final vectors c, r, the final probability is calculated by the following equation: $\Pr\{flag = 1 | c, r, M\} = \sigma(c^T Mr + b)$, [25], where the bias b and the matrix M are learned through the training of the model. If the probability is close to 1, it means it is more likely for the response to be correct for the specific context, while the probability being close to 0 means the opposite.

The training file, which was obtained through the preprocessing of the dataset, consists of triplets: context, utterance, flag. The use of context and utterance has been explained in the first paragraph of the model analysis. The flag, which gets a binary value (either 0 or 1), indicates if the response is the correct one for the specific context or the incorrect one. The model could possibly be trained just with the correct corresponding utterance for each context, but the inclusion of incorrect responses enhances the training. The goal is to minimize the loss function, which is the cross entropy loss function and incorporates the flag. The flag is usually within a logarithm, penalizing a prediction close to 1 if the response is an incorrect one or the opposite for a prediction close to 0 if the response is correct.

The following figure, taken from [25], showcases and summarizes the Siamese structure of the model:

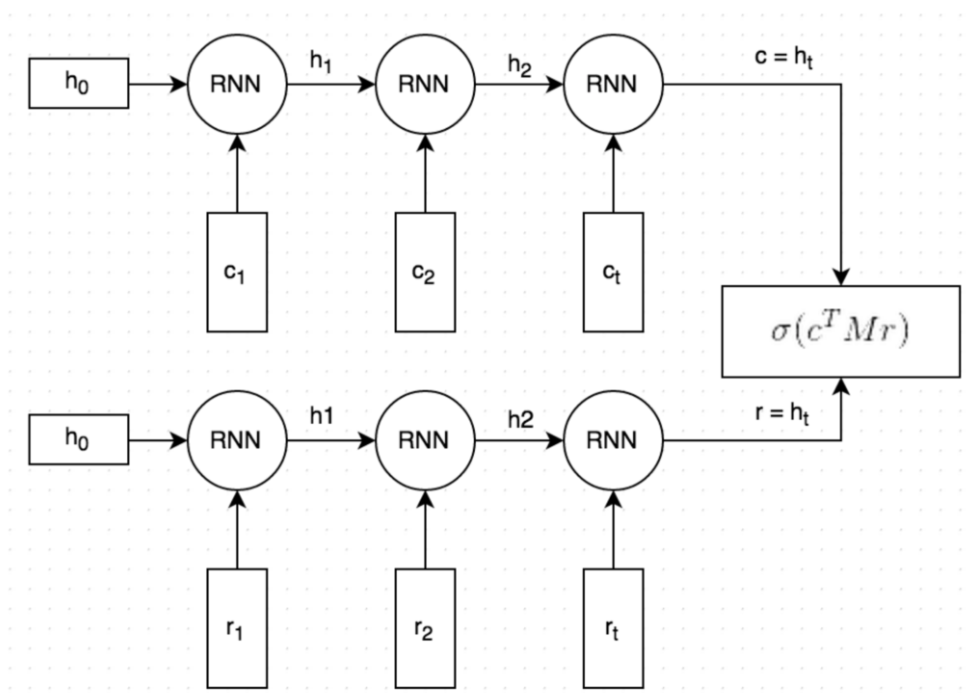

Figure 3-2: Siamese Neural Network architecture of the retrieval model

Evaluation of the model:

To evaluate the model, a modified version of **Recall@k** is used. For the evaluation, the preprocessed validation file of the dataset is used. It consists of 11 utterances for every context, 10 incorrect ones and 1 correct. Recall@k will be run through the same data for different k numbers. For any number ***k***, the model evaluates 11 responses for every context and then picks the best ***k*** ones. If the correct response is inside the ***k***-highest rated responses, then the model is perceived to be correct in that instance. When the final context is processed, a percentage of correct correspondences is calculated.

3.3 Generating a Response pool

After a retrieval model is trained and evaluated, how is it put to practical use? After all, the goal of a chatbot is to engage with the user in a dialogue, specialized or casual. During run-time, a retrieval chatbot receives as input the concatenated history of the (until that point) conversation and then ranks responses and selects the highest-ranked one to respond with. But, in order for a retrieval chatbot to be accurate, a vast and varied response selection is needed, most likely numbering in the millions. It is obvious then that a subset of responses is needed, due to the impossibility of the chatbot ranking the whole database of responses for every question. Curiously, this is a predominantly neglected part in the scientific body of work, seemingly because the interest lies in training the chatbot. Most endeavors have the chatbot rank a randomly selected subset of responses. However, this random response pool is inefficient, as it is possible that no appropriate responses will be selected. Therefore, the need arose for better and more dynamic response pool selection techniques and that will be the focus of the following paragraphs.

What we are looking for in all of the methods we will underline, is, **a higher probability** for the correct response to a question to be in our **candidate response pool**. For every context, we basically want to search through a list of responses that are more likely to contain the correct one. Whether it is through clustering or simpler frequency algorithms, they all converge to topic or response commonness. Based on the work of Swanson et al. [26], we highlight the two methods they used for the generation of the response pool and we propose a different and improved version of one method and a completely new one for producing a more efficient whitelist that will improve the compatibility of the response selection: The techniques will be: a) **Frequency-based**, b) **Clustering** and c) **topic-based**.

(**Note**: The experiments will use as a database of responses the isolated responses from the test file of the Ubuntu dataset. All the methods work, because every context has a single correct response associated with it in a 1:1 relationship. In that way, if the contexts are grouped, the corresponding responses associated with them are grouped as well.)

3.3.1 Frequency based

Mentioned in [26], the most straightforward method of generating a subset of the response database is to sort the responses based on selection frequency and select the top *n* (where *n* is a number of responses) as a response pool. The goal is to create a frequency hierarchy of most "valuable" responses, or to put it more accurately, most selected responses. Because the responses are finite and the retrieval model can only pick one out of predefined ones, it is evident that for numerous and lengthy runtimes of the chatbot, most responses will be chosen more than once, thus creating a hierarchy of frequencies. To provide an example, the response "Yes, you are correct" might have a very high frequency, as it fits many different contexts. Until the frequency hierarchy is created, at which point this technique can be used, the chatbot has to rank every response in the database when addressed with a question.

It would be easy to produce the frequency list in a situation where many users chatted with the chatbot model and background analytics calculated the frequency of the selected responses. If that is not a possibility, instead of users, the validation or test file of the dataset can be used. After the chatbot has been trained, the responses can be isolated and the test set can be processed through the chatbot and for every context in the test set, every response will be ranked resulting in a set of hierarchies of responses based on frequency. Then, a specific number of responses can be selected starting from the top, producing a response pool with a higher probability of having the correct response for a random context than a random response pool.

For a better understanding of this technique, let us use a more detailed hypothetical example. Let us say that we have a database of 7 responses in total and we want to select a response pool of just 3 responses for the chatbot to rank, instead of ranking all 7 for every question. We have had users chat with the chatbot and the total number of questions addressed to the chatbot were 53. That means that the chatbot answered 53 times, that is, that it chose one of the 7 responses in total 53 times. The results were the following:

Yes, you are correct.	20
I'm fine, you?	15
Not really...	13
Thank you!	3
How about it?	1
Maybe.	1
It's raining .	0

We can see that the model chose "Yes, you are correct" 20 times, "I'm fine you?" 15 times, etc… If we want to create a response pool with the 3 most frequent responses, then the pool will contain "Yes, you are correct", "I'm fine, you?" and "Not really…", because these were the 3 answers that were selected the most during runtime, and are more likely to be the logical responses to a question than a random response pool of 3 answers.

This method might not bring the most accurate response grouping, but it is an easy enough method to implement with straightforward results.

We do not provide our own implementation of this method, as we believe that more interest lies in the following techniques.

3.3.2 Clustering in a vector space

Swanson et al. [26] also experimented with clustering, by clustering all of the available **responses** in a **vector space** and then selecting the most frequent responses from each cluster. As mentioned above, one can create a hierarchy of frequencies by running many contexts (more than the responses) through the model. Swanson et al. combined the previous method with clustering for a more varied response pool. By selecting the most frequent responses from each cluster, they improve the first method by not just picking the most frequent responses from the whole database, but choosing the most frequent responses per “topic” or “meaning”, creating variety inside the response pool.

We aim to improve this method by using a **dynamic response pool** depending on the context, instead of a static one (that has the same pool for every context). What we propose, is clustering the contexts instead of the responses and when a new context arrives, we vectorize it too and find the cluster it belongs to and create a response pool by selecting the responses of the contexts inside the cluster. If the number of contexts-responses inside the cluster is not enough, we also select the responses of neighboring clusters (according to a distance metric). Our goal is to dynamically create a response pool that is approaching the “meaning” of the current context. Clustering can be performed in a vector space, after a suitable transformation of the contexts (e.g. Doc2Vec or in an Embedding layer, through their processing by a Neural Network).

To be more precise, we will follow two steps in order to create a Response Pool:

1. **Clustering of the contexts** using kmeans
2. **Selection of the responses** that correspond to the contexts inside the cluster of the arriving context or inside the neighboring clusters

3.3.2.1 Clustering the contexts through kmeans in a low-dimension vector space

In order to vectorize the contexts, we use Doc2Vec, a method by Mikolov and Le [34] which transforms the documents (contexts) into numerical representations (vectors). Doc2vec are neural network models that are included in *gensim.models* python libraries. Then, we cluster these vectors using kmeans. The test file is used, instead of the full training set which contains 1.000.000 lines of dialogs. The test file contains approximately 19.000 lines and is more manageable for demonstrating the method. Its structure is explained in 3.1. We isolate the contexts (and their responses) by choosing the first column from the respective data frame. The workflow of processing the data and getting a pool of responses to a certain context is shown in the block diagram of Figure 3-3.

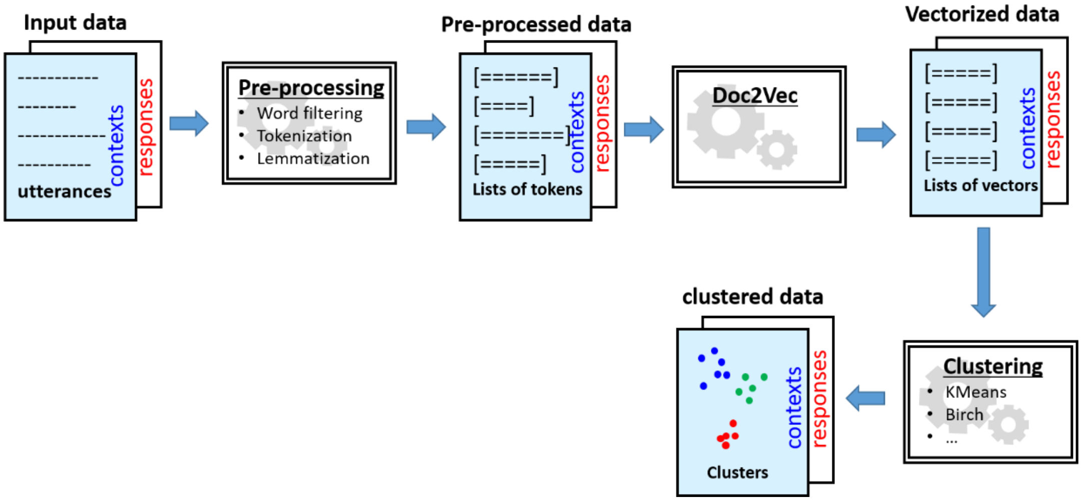

Figure 3-3: Workflow towards clusterizing the data in order to form a response pool

Specifically, the main steps of the clustering workflow and respective output samples are given below.

- A. Selection of the input data set: The test set, containing 18920 contexts with respective responses, is used.

- B. Pre-processing: Removing stop-words and other very common words from the input data set, like 'do', 'have', 'thank', etc.)

- C. Doc2Vec transformation into a low-dimensional vector space. The **contexts** are transformed into **vectors**. In our experiments we used a 5-D vector space, just for demonstration purposes. Example input data (c_docs, in a tokenized format), along with their output vectors(c_doc_vecs), are shown in Figure 3-4.

a/a	Contexts (c_docs)	Doc2Vec vectrors (c_doc_vecs)
0	['like', 'mean', 'guess', 'heck', 'bug', …]	[-0.08313535 0.00342988 -0.01575045 -0.0466449 -0.01139903]
1	['set', 'up', 'hd', 'such', 'type', 'passphras',…]	[0.00990149 0.06127585 -0.03118643 0.04714713 0.0580817]
2	['im', 'tri', 'ubuntu', 'macbook', 'pro', …]	[-0.06838371 -0.0178316 0.08604646 -0.00551041 0.0757286]
3	['suggest', 'link', 'remov', 'luk', 'passphras',..]	[0.05195832 -0.05883193 0.05944025 0.05467169 0.01203078]
4	['ad', 'second', 'usb', 'printer', 'sure', …]	[0.09222137 -0.01591114 0.01261725 -0.05538869 0.06121722]

Figure 3-4 : Output example of the Doc2Vec transformation

- D. Clustering through kmeans: The vectors are clustered using kmeans. Kmeans is a well-established popular technique for clusterizing vector data according to a distance metric (eg. Euclidean distance). Choosing a suitable **number of clusters** is an important issue, depending on the application. In this case and to showcase the concept, the algorithm is applied on the vectors derived from step C above, with ***k* = 3** clusters. As a result, each context is assigned to a cluster, which is identified by a label, as shown in Figure 3-5.

Out[65]:

	c_doc_vecs	c_docs	labels
0	[-0.08313535, 0.003429877, -0.015750453, -0.04...	[anyon, whi, stock, oneir, export, env, var, u...	2
1	[0.009901488, 0.061275847, -0.03118643, 0.0471...	[set, up, hd, such, type, passphras, access, b...	0
2	[-0.06838371, -0.017831597, 0.086046465, -0.00...	[im, tri, ubuntu, macbook, pro, read, forum, u...	1
3	[0.051958323, -0.058831926, 0.059440248, 0.054...	[suggest, link, remov, luk, passphras, boot, d...	1
4	[0.09222137, -0.015911141, 0.012617252, -0.055...	[ad, second, usb, printer, sure, should, read,...	2

Figure 3-5: Output example of the Doc2Vec transformation

Projection of the clusterized vectors on a 2-D space (after a Principal Component Analysis) gives the picture of Figure 3-6. For example, the third context in the table of Figure 3-5 (ordinal number 2) corresponds to the topic with label 1, so it is projected on a blue dot of the picture. All the clusters that are assigned to the topics with labels ‘0’, ‘1’ and ‘2’ form together the red, blue and green groups accordingly.

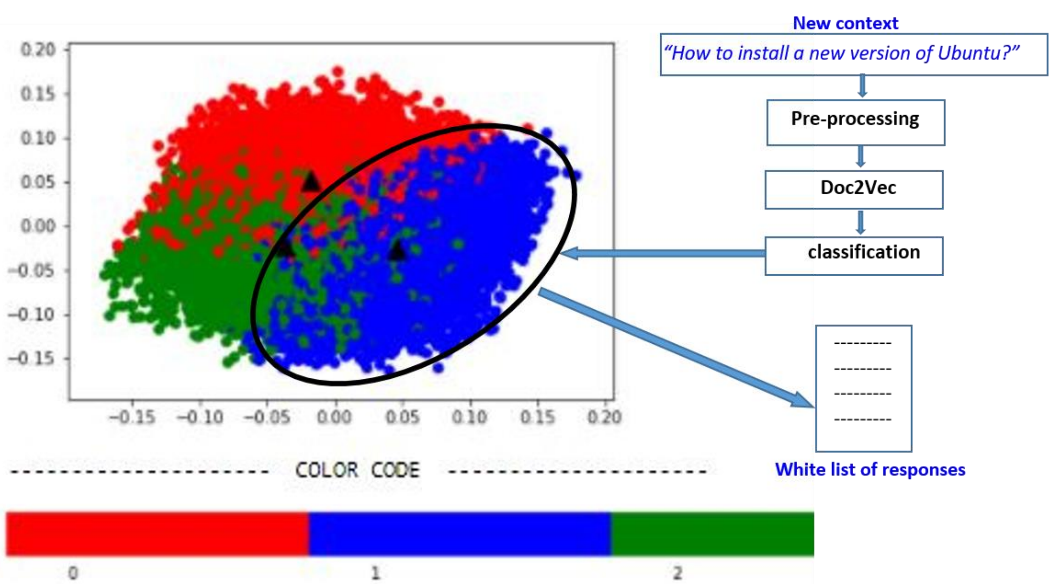

Figure 3-6: Clusters of the vectorized data set (3 clusters) projected on a 2-D space

3.3.2.2 Creating the Response pool through the clusters

Now that the contexts have been clustered, the technique is ready for the chatbot to dynamically create a response pool every time a new context arrives.

As shown in Figure 3-6, the procedure of generating a Response pool when a new context arrives is as follows: The new context is pre-processed in the same way as the dataset. Then, it is fed as input in the Doc2Vec model in order to be transformed into a vector. The vector is classified into the appropriate cluster (e.g. by choosing the nearest centroid), the contexts of which are used to derive the pool of responses.

To provide an example, let us say that the hypothetical context " How do I install Ubuntu? - What version do you use? – I am not sure" (which is the concatenated history of conversation) arrives. First, the context is preprocessed and what possibly remains is : [install, Ubuntu, version]. With Doc2vec the context is embedded to a vector, e.g. [0.01, 0.234, 0.123, 0.047, -0.2]. We figure out the distance from the centroids of the cluster, and conclude that it belongs to the blue cluster. Consequently, the response pool for this hypothetical example will be the responses that correspond to the contexts of the blue cluster.

As mentioned in the beginning, if the number of responses is desired to be specific , so as to not have too few or too many depending on how many contexts there in the cluster, the algorithm can be modified as follows: Until a certain threshold *n* is reached, where *n* the number of desired responses, we keep searching the neighboring clusters of the cluster that the arriving context belongs to.

3.3.2.3 Selection of Parameters (Dimensions, number of clusters) and Results of Clustering

Number of clusters :

In the previous showcase of the algorithm, we chose to use **3 clusters.** It is paramount to mention that 3 clusters for real life scenarios where response databases number in the millions would be too few to be able to provide a small and accurate response pool. However, because the test set was used, which consisted of approximately 19.000 responses (an already small sample), 3 clusters were adequate. Another reason 3 clusters were chosen, is because in different runs of the algorithm for higher numbers of clusters, the formed groups are not that distinct and bit muddled. The following figure shows the results of clustering with k = 16 clusters.

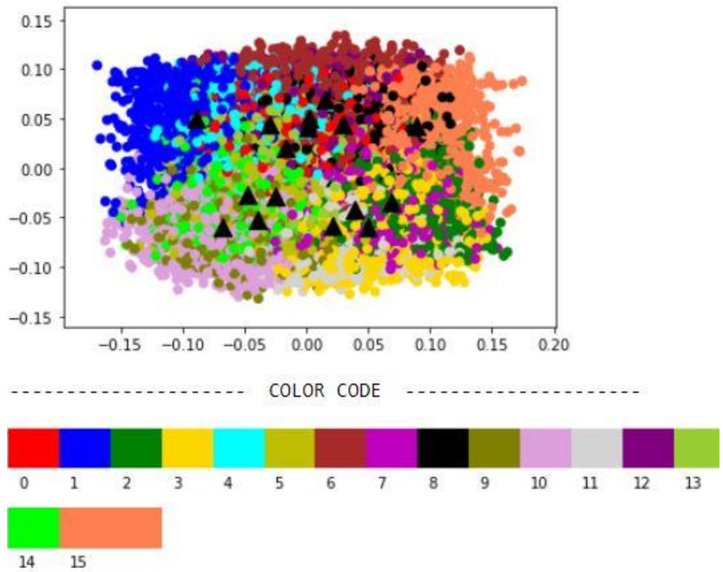

Figure 3-7: Clusters of the vectorized data set (16 clusters) projected on a 2-D space

Results and Conclusions:

This clustering technique aims to deliver a dynamic response pool for every new context. Swanson et al. use clustering to generate a varied response pool of frequent responses, but once it is created, it will remain the same no matter the arriving context.

The results of clustering with k = 3 clusters are 3 distinct groups, which means that vectors in a cluster have some sort of arithmetic similarity, which hopefully leads to similarity of meaning. Instead of searching through the whole database, the size of the response pool can be reduced to 1/3 and still maintain high accuracy. More experiments can be carried out to figure out the optimal number of clusters and size of the response pool.

3.3.3 Topic-based classification

With the aforementioned clustering method, a different response pool can be created for each context based on the proximity of vectors. That does not guarantee the topic similarity of adjacent clusters, only that they have arithmetic similarity during their vectorization (that highly depends on the vectorization or embedding method) and that is the key difference between the previous method and this one.

Therefore, to further improve the generation of a dynamic response pool, we propose a new method which finds out the most important **topics** of the corpus using **Latent Dirichlet Allocation (LDA)**, a method used to find hidden semantic structure in documents [30]. (It is a type of clustering in a probability distribution space that groups documents around hidden topics/meanings. It will be explained further in 3.3.3.1.)

The idea behind this method is to improve the accuracy of the response pool and increase the probability of the correct response being in the response pool by finding the most important **hidden topics** inside the corpus of the data and figuring out the relevance between the arriving context and the topics. There are keywords or thematic resemblance of the context that can connect it to any one of these topics and then all that is left to do, is search for responses of contexts within that particular group or groups. It is, therefore, quite similar to the clustering method, only that now there are topics instead of clusters that function in a similar way.

As before, there are two main steps to this method:

1) Applying **LDA** to the entirety of the documents (contexts) to find hidden topics within them and grouping the contexts according to their most contributing topic.
2) Creating the **Response pool** by selecting the responses of the contexts that belong to the most contributing topics of the arriving context.

Note: Similar to the clustering method, the test set will be used, which contains almost 19.000 lines of dialogue, instead of the training set, for demonstration purposes.

3.3.3.1 Applying Latent Dirichlet Allocation to the documents to find the hidden topics

Latent Dirichlet Allocation:

Latent Dirichlet Allocation (LDA) is a technique first proposed in 2000 and revisited in 2003 and in 2012 (in "probabilistic models") by Blei et al. It is a "generative probabilistic model of a corpus" [30]. According to LDA, each document is a mixture of a number of **hidden topics**. A topic is defined as a "**distribution** over a fixed vocabulary" [30]. For example, a topic regarding geography could be the following distribution: 0.2"river" + 0.3"mountain" + 0.4"city" + 0.1"people". These topics are blended in different proportions. Exactly how it is showcased in the previous example, a topic regarding geography will have words related to geography in high probabilities. Given a document, LDA backtracks and tries to surmise what hidden topics would create this document by using statistical inference techniques. It is like reversing the process of generation by going through the documents and thinking what topics could produce those documents.

The application of LDA in our case is shown diagrammatically in Figure 3-8:

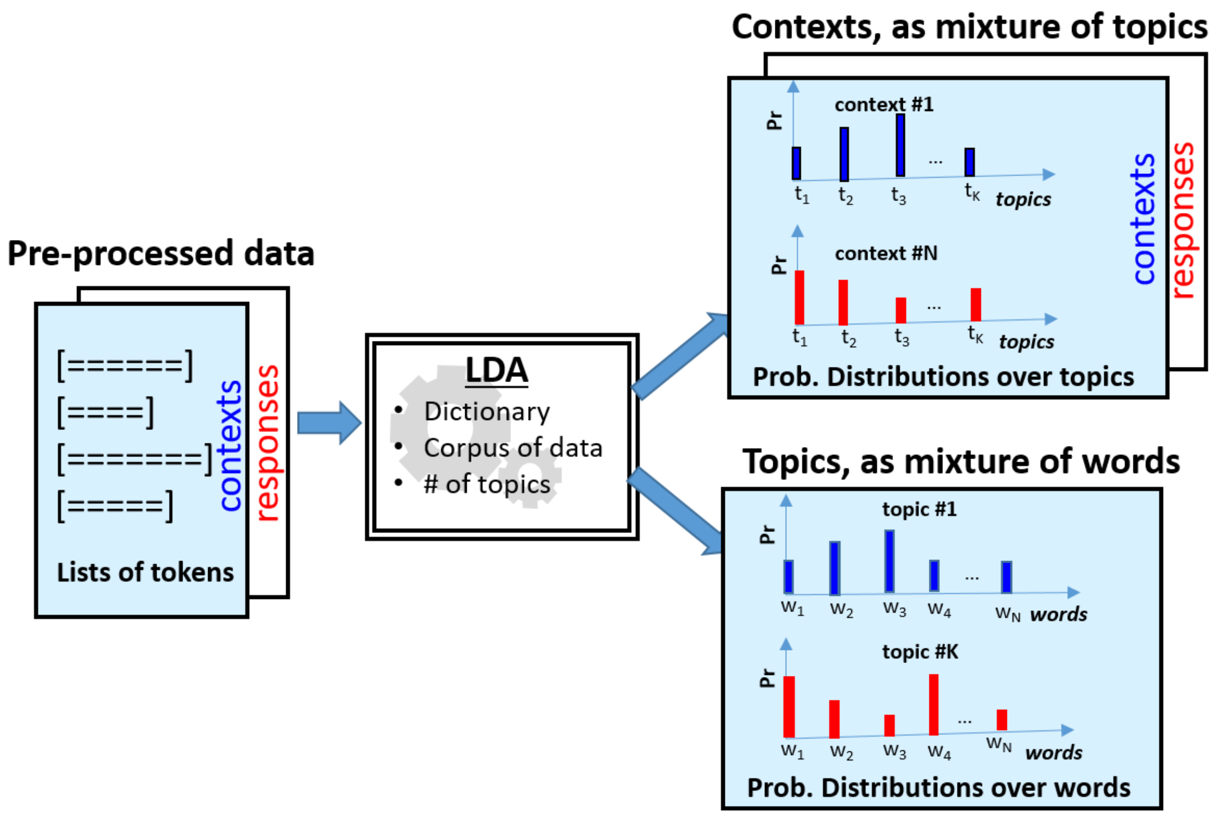

Figure 3-8: Finding topics in the corpus of contexts through LDA

We implemented the LDA process in python by using the gensim libraries, closely following the work and instructions of Susan Li from Towards Data Science [31] and applying it to our case and ran the test file from the dataset through it, according to the following steps:

A. <u>Selection of Input Data and Pre-processing:</u> It is much the same as in the clustering method (test set as input data and removing stop-words or very common words, tokenizing, …), so steps A and B from before are merged.

B. <u>Selection of the number of topics</u>: Since the number of topics is not fixed and known beforehand, the best way to decide on the number would be to make successive runs by increasing the number of topics and estimating the quality of the outcome in each run. However, to better showcase the method and just like in the clustering method (where 3 clusters were chosen), we chose just **3 topics** to highlight the distinction between the topics.

C. <u>Word distribution for each topic</u>. In this step, the 3 most dominant hidden topics are found. As mentioned before, every topic is characterized by a certain mixture of words. Figure 3-9 shows the 3 discovered topics as mixtures of words (the 10 most significant words are shown for each topic). We can see, for example, that the first topic is a mixture of 0.019 of the word “ubuntu”, 0.018 of the word “install” and so on… This combination of percentages singularly describes this particular topic. Because it is not intuitive to use the sum of percentages as an identifier for the topic, we will just use labels ‘0’, ‘1’ and ‘2’.

```
# Print the Keyword in the topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0,
  '0.019*"ubuntu" + 0.018*"instal" + 0.013*"work" + 0.012*"tri" + '
  '0.011*"window" + 0.011*"boot" + 0.010*"like" + 0.009*"when" + 0.009*"up" + '
  '0.009*"doe"'),
 (1,
  '0.043*"file" + 0.017*"partit" + 0.016*"command" + 0.016*"user" + '
  '0.015*"mount" + 0.013*"drive" + 0.013*"sudo" + 0.011*"dev" + 0.011*"root" + '
  '0.010*"need"'),
 (2,
  '0.028*"get" + 0.023*"ubuntu" + 0.023*"instal" + 0.016*"http" + 0.013*"tri" '
  '+ 0.012*"com" + 0.012*"apt" + 0.010*"packag" + 0.010*"driver" + '
  '0.009*"work"')]
```

Figure 3-9: Mixture (distribution) of words for each topic (the 10 most significant)

D. Topic distribution for each context. Similarly, each context is described as a combination of topics. The table in Figure 3-10 shows the most contributing topic for each context. In this particular excerpt with the first 20 contexts the topic labeled as 0 prevails.

	Context_No	Dominant_Topic	Topic_Perc_Contrib	Topic_Keywords	Context
0	0	0.0	0.4098	ubuntu, instal, work, tri, window, boot, like,...	[anyon, whi, stock, oneir, export, env, var, u...
1	1	0.0	0.4919	ubuntu, instal, work, tri, window, boot, like,...	[set, up, hd, such, type, passphras, access, b...
2	2	0.0	0.5275	ubuntu, instal, work, tri, window, boot, like,...	[im, tri, ubuntu, macbook, pro, read, forum, u...
3	3	0.0	0.5101	ubuntu, instal, work, tri, window, boot, like,...	[suggest, link, remov, luk, passphras, boot, d...
4	4	2.0	0.5976	get, ubuntu, instal, http, tri, com, apt, pack...	[ad, second, usb, printer, sure, should, read,...
5	5	0.0	0.4389	ubuntu, instal, work, tri, window, boot, like,...	[suggest, link, question, sorri, idea, disabl,...
6	6	0.0	0.5074	ubuntu, instal, work, tri, window, boot, like,...	[look, mean, look, program, im, sure, anyth, b...
7	7	0.0	0.7414	ubuntu, instal, work, tri, window, boot, like,...	[probabl, someth, stupid, ca, figur, instal, u...
8	8	2.0	0.4344	get, ubuntu, instal, http, tri, com, apt, pack...	[hey, guy, tri, write, script, need, network, ...
9	9	0.0	0.5945	ubuntu, instal, work, tri, window, boot, like,...	[instal, ubuntu, laptop, notic, get, poor, bat...
10	10	2.0	0.7906	get, ubuntu, instal, http, tri, com, apt, pack...	[even, need, some, java, chrome, keep, get, er...

Figure 3-10 : The most contributing topic for each context

E. Grouping the contexts into topics. With LDA, we have found the 3 hidden topics, as shown in Figure 3-9. Before the response pool for an arriving context can be compiled, all the contexts must first be **grouped** all the contexts into the 3 topics. We do that by placing the context in the group of its most contributing topic. For example, a context in the dataset might have 80% contribution from the '1' topic. In that case, it can be easily placed into the '1' group. This grouping works, if the topics are somewhat distinct. If in the distribution of the topics, the percentages are similar (e.g. 24% of the first topic and 30% of the second), this method might be problematic. It can be improved by placing the context into both groups and then modifying the selection method accordingly. However, for just three topics this is not an issue, as they are quite distinct and away from each other. This is presented clearly in Figure 3-11.

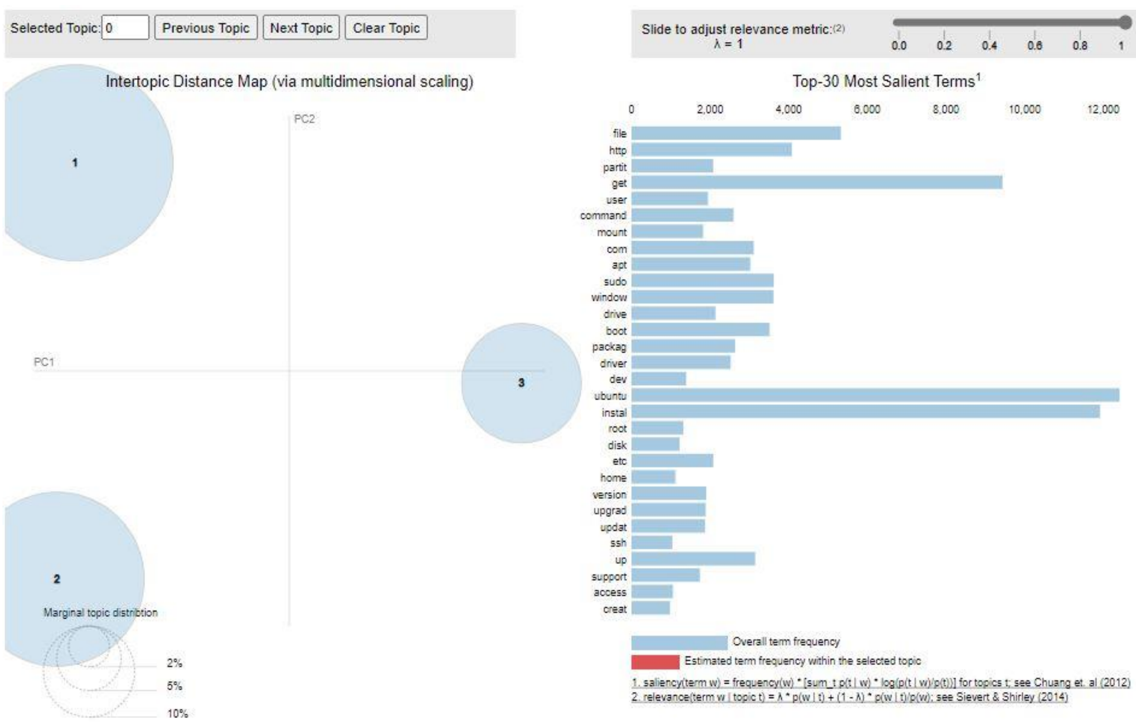

Figure 3-11 : Visualization of topic distancing and their probability distribution over words (3 topics)

3.3.3.2 Compile a Response pool

For the next step and, mirroring the clustering method, we compile a **Response Pool** for a new arriving context. When a new context arrives, it is first preprocessed and, then, its topic distribution (mixture) is derived by using the trained LDAmodel. From this, the most important topic associated with the context is found and, by virtue of step E, all the contexts associated with it. If it is a very large topic, only the most probable contexts-responses from the group are selected to match the desirable number of responses in the response pool. If it is a small topic, all the responses from the topic are selected and we move on to the next important topic associated with the context until we match the appropriate number of responses.

3.3.3.3 Parameters and Results of LDA

Number of topics :

In the example above and just like the clustering method, **3 topics** were chosen for the sake of better understanding the method. However, a higher number of topics might be more desirable especially for a larger set than the test set. If 16 topics instead of 3 are chosen, the distribution of topics and their distance would be portrayed as shown in the following figure:

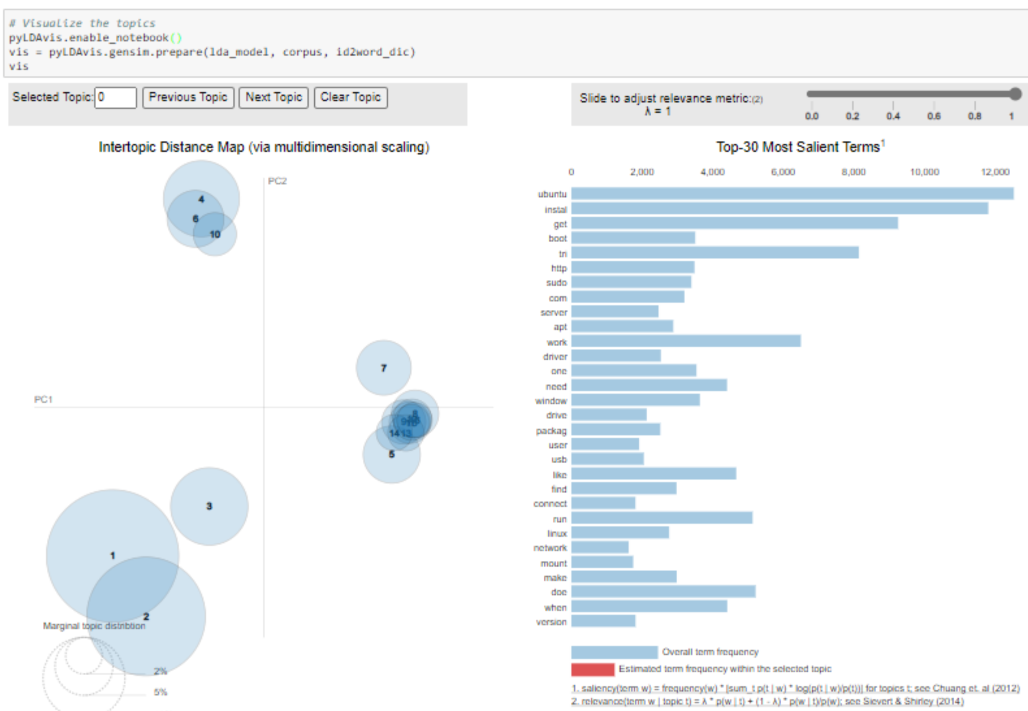

Figure 3-12 : Visualization of topic distancing and their probability distribution over words (16 topics)

As we can see, some topics, just like with the 16 clusters (in the previous kmeans clustering method), are muddled and very close together. We can see, however, that 3 main groups of topics are formed. That means that these 3 groups have some relevance. The distance of the topics correlates to thematic distance. Further experiments and runs need to be carried out to figure out the best grouping of the topic for optimal efficiency of the response pool and the chatbot.

Results-Conclusion :

The results of the LDA method, much like the clustering method, can significantly reduce the size of the response pool (depending on the number of topics and how we choose to select responses from each group) and even more accurately portray the “meaning” of the arriving context in its most contributing topic.

Another point of interest and a benefit of the LDA analysis in comparison to the clustering method, is that the resulting topics are **human interpretable**. That means, that we can visualize the topics and understand their substance. We can see, in the 3 topic example, all the words of high importance for the topic and understand what the topic is about. For example, the third topic with the label ‘2’ has words like “http”, “apt” and “com” and we can very easily surmise that the topic is network-related, something not possible with the clustering method. That is extremely helpful for the evaluation and improvement of the method and, subsequently, the evaluation and improvement of the chatbot.

More experiments need to be done to prove the technique’s usefulness.

3.3.4 Summary of Response Pool Generation techniques

It is apparent that a lot of work can be put into creating appropriate and efficient response pools in order to minimize the chatbot’s response delay and maximize response accuracy. While the focus of the thesis remains on the preprocessing and grouping of data and not in proving the usefulness of those techniques for the chatbot in action, it is self-evident that the reduction of the pool size and the increase of the relevance of responses can only be beneficial to any chatbot.

Random selection is lacking and the **frequency based** technique improves the random selection method, but it still creates a static response pool with a high chance of inaccuracies. The **clustering of contexts**, instead of clustering of responses, results in the generation of a dynamic response pool. The arriving context will only search the cluster it belongs to or the neighboring clusters and select their responses as a response pool. This can improve on the static clustering response pool method employed by Swanson et al, although more research needs to be carried out to figure out the optimal number of clusters and size of the response pool. As a continuation of this method, instead of clustering, **Latent Dirichlet Allocation** can be used to group according to hidden topics of the entirety of the contexts. For any arriving context, we can extract the percentage of topics that compose the context and then use as a response pool the responses inside the group of the context’s dominating topics. The main benefit of this method, is that it can group the corpus into distinct human interpretable topics, instead of vague arithmetic clusters. The hidden topics that were found for the dataset were promising, as they were common topics one would expect to find in a technical forum.

The foundation for the clustering and LDA methods are set and are promising, but more research on more datasets needs to be done together with chatbot experimentations for the results to be conclusive.

4 CONCLUSIONS

Chatbots have evolved drastically throughout the years. From humble beginnings, where researchers and psychologists have tried to impersonate a psychologist or a patient using a computer, chatbots have now evolved mostly through the use of neural networks and Machine Learning and are capable of important and useful contributions to many aspects of daily life. Looking at their history, one can see their visible progress with every major milestone (Eliza, Parry, Dr. Sbaitso, etc..), especially this last decade with their expansion to commercial products like Siri and Alexa.

The biggest contribution of chatbots remains in customer support and virtual assistants, but important research shows their potential in education and mental health. Especially during the period of time of this thesis that the coronavirus lockdown throughout the globe has forced schools and universities to adopt remote strategies for teaching and exams, the benefits of chatbots shown by research should be all the more enticing for academics. Making use of the information retrieval capabilities of chatbots, the teacher's workload can be significantly reduced and the students' engagement can even be increased. On the topic of mental health, conversational agents have been shown to help patients open up and be more comfortable conversing with them than with a clinical psychologist, an indication of which was shown since 1966 with the Eliza chatbot. Although research has been fairly clear on the advantages of chatbots on these two specific areas, their implementation has been lacking compared to customer support and virtual assistants. It is of a high importance that more research on the usefulness of chatbots on education and mental health is done, so that the future will show greater use of their strengths in these areas.

The classification of chatbots by the range of topics that can adapt to and by their implementation design is also explored. Conversational agents can be categorized into task-oriented and open domain, as well as rule-based, retrieval and generative. The familiarity with these categories is integral, in order for a chatbot to be most effective for a specific role. An open domain generative chatbot is effective in different roles than a task-oriented rule-based one.

Another point of interest is the evaluation of chatbots, which can be categorized into Content Evaluation and User Experience Evaluation. The first is mostly carried out by computers using specific techniques like Recall@k and the second by humans evaluating their satisfaction using the chatbot.

Instead of training our own model, we relied on the model of Lowe et al. [24] to focus on generating a **response pool for the retrieval model**, a partially neglected part of the scientific body of research. After experimenting with clustering and grouping the responses on hidden topics using LDA, our conclusions were that even with a very specific and scientific dataset like the one used from [24], it is still possible to have good results in clustering and in applying LDA because the documents are grouped around technical terms. The LDA method, in particular, can group the documents into human interpretable topics, which can lead to much better evaluation and improvement of the response pool selection. While more and more applications are built around generative models, there is still important research and work to be done for retrieval models and the generation of their response list going forward.

ABBREVIATIONS - ACRONYMS

NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
LDA	Latent Dirichlet Allocation
FAQ	Frequently Asked Questions
Q&A	Question and Answer

REFERENCES

[1] Business Insider (2016). 80% of businesses want chatbots by 2020 (https://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12)

[2] Michael L. Mauldin (1994). Chatterbots, TinyMuds and the Turing Test : Entering the Loebner

Prize Competition. AAAI '94, ISBN : 0262611023

[3] Oppy, Graham and David Dowe, "The Turing Test", The Stanford Encyclopedia of Philosophy (Winter 2020 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2020/entries/turing-test/>.

[4] Ioana Alina Cristea, Madalina Sucala, Daniel David (2013). Can you tell the difference? Comparing face-to-face versus computerbased interventions. The "Eliza" effect in psychotherapy. Journal of Cognitive and Behavioral Psychotherapies 13(2):291-298

[5] Shum, H., He, X. & Li, D (2018). From Eliza to XiaoIce: challenges andopportunities with social chatbots. Frontiers Inf Technol Electronic Eng 19, 10–26.

[6] Tomáš Zemčík (2019). A Brief History of Chatbots. DEStech Transactions on Computer Science and Engineering, ISBN: 978-1-60595-643-5

[7] B. A. Shawar and E. Atwell (2015), "ALICE chatbot: Trials and outputs," Comput. y Sist., vol. 19, no. 4, pp. 625–632, doi: 10.13053/CyS-19-4-2326

[8] SAFEATLAST (2019). Is Siri Better Than Google? – 16 Interesting Siri Statistics. (https://safeatlast.co/blog/siri-statistics/)

[9] Winkler, R., Söllner, M. (2018).: Unleashing the Potential of Chatbots in Education : A State-OfThe-Art Analysis . In : Academy of Management. Meet. Annu. Chicago, A O M.

[10] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, Art Pettigrue (2018). Conversational AI: The Science Behind the Alexa Prize, arXiv:1801.03604 [cs.AI].

[11] Greer S, Ramo D, Chang Y, Fu M, Moskowitz J, Haritatos J (2019)

Use of the Chatbot "Vivibot" to Deliver Positive Psychology Skills and Promote Well-Being Among Young People After Cancer Treatment: Randomized Controlled Feasibility Trial

JMIR Mhealth Uhealth;7(10):e15018, doi : 10.2196/15018

[12] Robbie Gonzalez (WIRED 2017). Virtual Therapists Help Veterans Open Up About PTSD (https://www.wired.com/story/virtual-therapists-help-veterans-open-up-about-ptsd/)

[13] Vaidyam, A. N., Wisniewski, H., Halamka, J. D., Kashavan, M. S., & Torous, J. B. (2019). Chatbots and Conversational Agents in Mental Health: A Review of the Psychiatric Landscape. *The Canadian Journal of Psychiatry*, *64*(7), 456–464. https://doi.org/10.1177/0706743719828977

[14] S. Mujeeb, M. H. Javed and T. Arshad (2017), "Aquabot: A Diagnostic Chatbot for Achluophobia and Autism," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 8, no. 9, pp. 209-216

[15] Winkler, R. & Söllner, M. (2018): Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis. In: Academy of Management Annual Meeting (AOM). Chicago, USA.

[16] Bob Heller, Mike Proctor, Dean Mah, Lisa Jewell, and Bill Cheung(2005). Freudbot: An investigation of chatbot technology in distance education. In EdMedia+ Innovate Learning. Association for the Advancement of Computing in Education (AACE), 3913–3918

[17] G. Molnár and Z. Szüts (2018), "The Role of Chatbots in Formal Education," IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2018, pp. 000197-000202, doi: 10.1109/SISY.2018.8524609.

[18] Roos, S. (2018). Chatbots in education : A passing trend or a valuable pedagogical tool? (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-355054

[19] Hussain S., Ameri Sianaki O., Ababneh N. (2019) A Survey on Conversational Agents/Chatbots Classification and Design Techniques. In: Barolli L., Takizawa M., Xhafa F., Enokido T. (eds) Web, Artificial Intelligence and Network Applications. WAINA 2019. Advances in Intelligent Systems and Computing, vol 927. Springer, Cham. https://doi.org/10.1007/978-3-030-15035-8_93

[20] Jintae Kim, Hyeon-Gu Lee, Harksoo Kim, Yeonsoo Lee, Young-Gil Kim (2018). Two-Step Training and Mixed Encoding-Decoding for Implementing a Generative Chatbot with a Small Dialogue Corpus. Association for Computational Linguistics, doi: 10.18653/v1/W18-6707

[21] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Association for Computational Linguistics, doi: 10.3115/v1/D14-1179

[22] Wari Maroengsit, Thanarath Piyakulpinyo, Korawat Phonyiam, Suporn Pongnumkul, Pimwadee Chaovalit, and Thanaruk Theeramunkong (2019) . A Survey on Evaluation Methods for Chatbots. In Proceedings of the 2019 7th International Conference on Information and Education Technology (Aizu-Wakamatsu, Japan) (ICIET 2019). Association for Computing Machinery, New York, NY, USA, 111–119. https://doi.org/10.1145/3323771.3323824

[23] Inbenta (2020), 10 Key Metrics to Evaluate your AI Chatbot Performance (https://www.inbenta.com/en/blog/10-key-metrics-to-evaluate-your-ai-chatbot-performance/)

[24] Weixin Liang, James Zou, Zhou Yu (2020). Beyond User Self-Reported Likert Scale Ratings: A Comparison Model for Automatic Dialog Evaluation. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, doi : 10.18653/v1/2020.acl-main.126

[25] Ryan Lowe, Nissan Pow, Iulian Serban, Joelle Pineau (2015). The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. Association for Computational Linguistic, doi : 10.18653/v1/W15-4640.

[26] Kyle Swanson, Lili Yu, Christopher Fox, Jeremy Wohlwend, Tao Lei (2019). Building a Production Model for Retrieval-Based Chatbots. Association for Computational Linguistics, doi : 10.18653/v1/W19-4104.

[27] Caros M. (2019). A Generative Dialogue System for Reminiscence Therapy, Master thesis of the MET program at UPC ETSETB TelecomBCN.

[28] Cristian Danescu-Niculescu-Mizil, Lilian Lee (2011). Cornell Movie-Dialogs Corpus, Cognitive Modeling and Computational Linguistics Workshop at ACL 2011, Cornell University.

[29] Rudol Kadlec, Ubuntu Dialogue Corpus v2.0 (rkadlec on Github) (https://github.com/rkadlec/ubuntu-ranking-dataset-creator)

[30] David M. Blei (2012). Probabilistic Topic Models. Communications of the ACM, doi: 10.1145/2133806.2133826.

[31] Susan Li (2018), Topic Modelling in Python with NLTK and Gensim, from Towards Data Science: https://towardsdatascience.com/topic-modelling-in-python-with-nltk-and-gensim-4ef03213cd21.

[32] Sundermeyer, Martin / Schlüter, Ralf / Ney, Hermann (2012): "LSTM neural networks for language modeling", In INTERSPEECH-2012, 194-197.

[33] Michael Pi (2018). Illustrated Guide to Recurrent Neural Networks, from Towards Data Science: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

[34] Quoc V. Le, Tomas Mikolov (2014). Distributed Representations of Sentences and Documents. Publisher: PMLR, Conference: International conference on machine learning.

[35] Deriu, J., Rodrigo, A., Otegi, A. et al. Survey on evaluation methods for dialogue systems. Artif Intell Rev 54, 755–810 (2021). https://doi.org/10.1007/s10462-020-09866-x