CHATBOT: Architecture, Design, & Development

By Jack Cahn

Thesis Advisor: Dr. Boon Thau Loo
Engineering Advisor: Dr. Jean Gallier

Senior Thesis (EAS499)
University of Pennsylvania
School of Engineering and Applied Science
Department of Computer and Information Science
April 26, 2017

Table of Contents

1. Introduction

Chatbots are “online human-computer dialog system[s] with natural language.” [1] The first conceptualization of the chatbot is attributed to Alan Turing, who asked “Can machines think?” in 1950. [3] Since Turing, chatbot technology has improved with advances in natural language processing and machine learning. Likewise, chatbot adoption has also increased, especially with the launch of chatbot platforms by Facebook [93], Kik [94], Slack [95], Skype [96], WeChat [97], Line [98], and Telegram [99]. By September 2016, Facebook Messenger hosted 30,000 bots and had 34,000 developers on its platform. [100] The Kik Bot Shop announced in August 2016 that the 20,000 bots created on its platform had “exchanged over 1.8 million messages.” [101]

This paper is a literature review of the design choices, architecture, and algorithms used in chatbots. Section 1 will describe chatbot function and history in more detail and discuss the methods used to evaluate chatbots. Section 2, will walk through chatbot functionality step-by-step, beginning with automatic speech recognition (ASR) algorithms, natural language processing (NLP) functionality, response generation approaches, knowledge base creation strategies, and dialogue management (DM) algorithms, and concluding with a discussion of text to speech algorithms. Section 3 will focus on chatbot development, beginning with a case study of IBM Watson’s chatbot functionality and concluding with a discussion of security considerations and chatbot applications.

2. Chatbot Overview

2.1. Background & History

2.1.1. What is a Chatbot?

A chatbot is an "online humn-computer dialog system with natural language." [1] Sansonnet et. al. [2] provides a basic framework which outlines the functions expected from modern chatbots:

A. Dialogic Agent: must understand the user, i.e. provide the function of comprehension. Bots are provided with a textual (or oral, see Section 2.1) input, which are analyzed with natural language processing tools, and used to generate appropriate responses. [2]

B. Rational Agent: must have access to an external base of knowledge and common sense (e.g. via corpora of data) such that it can provide the function of competence, answering user questions. Should store context-specific information (e.g. user's name, etc.). [2]

C. Embodied Agent: should "provide the function of presence…once regarded as very optional…this function proves to be crucial in the case of ordinary users." Even the earliest bots were given names (ELIZA, ALICE, CHARLIE, etc.) in order to satisfy this condition. Today, developers are focused on the use of language tricks to create personas for chatbots in order to build trust with users and give the impression of an embodied agent. [2]

2.1.2. Chatbot History

In 1950, Alan Turing asked the question "Can machines think?" Turing conceptualized the problem as an "imitation game" (now called the Turing Test), in which an "interrogator" asked questions to human and machine subjects, with the goal of identifying the human. If the human and machine are indistinguishable, we say the machine can think. [3] In 1966, Joseph Weizenbaum at MIT created the first chatbot that, arguably, came close to imitating a human: ELIZA. Given an input sentence, ELIZA would identify keywords and pattern match those keywords against a set of pre-programmed rules to generate appropriate responses. [4]

Since ELIZA, there has been progress in the development of increasingly intelligent chatbots. In 1972, Kenneth Colby at Stanford created PARRY, a bot the impersonated a paranoid schizophrenic. [5] In 1995, Richard Wallace created A.L.I.C.E, a significantly more complex bot that generated responses by pattern matching inputs against <pattern> (input) <template> (output) pairs stored in documents in a knowledge base. These documents were written in Artificial Intelligence Markup Language (AIML), an extension of XML, which is still in use today. ALICE is a three-time winner of the Loebner prize, a competition held each year which attempts to run the Turing Test, and awards the most intelligent chatbot. [6]

Modern chatbots include: Amazon's Echo and Alexa, Apple's Siri, and Microsoft's Cortana. [7] The architectures and retrieval processes of these bots take advantage of advances in machine learning to provide advanced "information retrieval" processes, in which responses are generated based on analysis of the results of web searches. Others have adopted “generative” models to respond; they use statistical machine translation (SMT) techniques to “translate” input phrases into output responses. Seq2Seq, an SMT algorithm that used recurrent neural networks (RNNs) to encode and decode inputs into responses is a current best practice. [8]

2.2. Evaluation

2.2.1. Evaluation Perspectives

There are a number of different perspectives on how to evaluate chatbot performance. From an information retrieval (IR) perspective, chatbots have specific functions: there are virtual assistants, question-answer and domain-specific bots. Evaluators should ask questions and make requests of the chatbot, evaluating effectiveness by measuring accuracy, precision, recall, and F-score relative to the correct chatbot response. [9]

From a user experience perspective, the goal of the bot is, arguably, to maximize user satisfaction. Evaluators should survey users (typically, measured through questionnaires on platforms such as Amazon Mechanical Turk), who will rank bots based on usability and satisfaction. From a linguistic perspective, bots should approximate speech, and be evaluated by linguistic experts on their ability to generate full, grammatical, and meaningful sentences. [9]

Finally, from an artificial intelligence perspective, the bot that appears most convincingly human (e.g. passes the Turing Test best) is the most effective. [9]

2.2.2. PARADISE Framework

One of the most widely used frameworks which fuses these perspectives is the PARAdigm for DIalogue System Evaluation (PARADISE). First and foremost, PARADISE estimates subjective factors such as: (i) ease of usage, (ii) clarity, (iii) naturalness, (iv) friendliness, (v) robustness regarding misunderstandings, and (vi) willingness to use the system again. It does so by collecting user ratings through the distribution of questionnaires. Second, PARADISE seeks to objectively quantify bot effectiveness by maximizing task success and minimizing dialogue costs. [10]

Maximizing Task Effectiveness: To maximize task effectiveness, bots must deliver correct results (i.e. the information retrieval perspective). Walker et. al., the creators of PARADISE, introduce the concept of an attribute value matrix (AVM) to measure effectiveness. They propose human subjects follow a script with a bot, which is supposed to achieve certain desired outcomes (e.g. the generation of informative responses). These “correct” responses are coded into a scenario key. Meanwhile, the results of the bot being tested are coded into the AVM. [10]

From the AVM and scenario key, a confusion matrix M is created, which encodes the number of times the bot retrieved the right vs. wrong answer. [10] To measure task success, Walker et. al. compute the kappa coefficient of the bot based on the confusion matrix:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \text{ [10]}$$

P(A) is the proportion of time where the AVM agrees with the correct result from the scenario key, and P(E) is the probability of agreement by chance; a bot generating random answers would have $\kappa = 0$ whereas a human would have $\kappa = 1$. [10]

Bates et. al. argue that the task effectiveness metric is too punitive: it considers only one correct answer, and reduces the score for bots that do not reach this answer. Instead a “Dialogue Breadth Test” should be run, in which 10 answers are accepted to the first utterance, and 10 answers accepted at the next turn (e.g. 100 overall acceptable second utterances). Such a modification would allow the system to better capture acceptable variability. [11]

Minimizing Dialogue Costs: Walker et. al. define two types of dialogue costs: efficiency costs and qualitative costs. Efficiency cost metrics include: (i) total elapsed time, (ii) total number of system turns, (iii) total number of system turns per task, and (iv) total elapsed time per turn. Qualitative cost metrics include the: (i) number of re-prompts, (ii) number of user barge-ins, (iii) number of inappropriate system responses, (iv) concept accuracy, and (v) turn correction ratio. [12]

Overall performance is the difference between task effectiveness and costs. Specifically:

$$Performance = \alpha * N(\kappa) - \sum_{i=1}^{n} w_i * N(c_i) \text{ [10]}$$

α is a weight on k, each cost function is weighted by w_i, and N is a Z score normalization function. [10]

2.2.3. Other Evaluation Methods

There exist a wide variety of evaluation methods outside of the PARADISE framework which are used to evaluate the performance of commercially-available chatbots. Kuligowska et. al., for example, propose an evaluation framework which they apply to 29 polish-speaking chatbots. [102]

First, Kuligowska et. al. measure the “visual look” of the chatbot, rewarding bots that resemble living people (e.g. fulfill the embodied agent requirement). [102] Van Vugt et al. found in 2010 that bots that appear human increase user engagement and the willingness of individuals to interact with the bot. [103] Using this approach, bodiless chatbots such as PayU’s Wirtualny Doradca and cartoon-like bots such as IKEA’s Ania are penalized. [102]

Second, Kuligowska et. al. evaluate the implementation of the chatbot system on its host website. To receive a 5/5 rating, websites had to implement both a built-in window users could use to interact with the bot, and a pullout side tab which users could toggle when they wanted to talk to the bot. Websites with only a pullout tab receive a rating of 4/5, while those with only a fixed built-in window received a rating of 3/5. [102]

Third, Kuligowska et. al. evaluate the ability of each bot to produce speech. Bots with unique custom voices were rated as 5/5, chatbots with standard text-to-speech modules but without a custom voice module received a 4/5, those without shutdown options received a 3/5, and those chatbots that could only communicate via text not speech received a 1/5 rating. [102]

Fourth, Kuligowska et. al.'s framework sought to assess the knowledge base of each bot, both in terms of general knowledge, and domain-specific knowledge relevant to the bot's website (e.g. IKEA's chatbot should have knowledge of IKEA products). To assess general knowledge, Kuligowska et. al. asked each bot a set of standard general knowledge questions, and assigned binary scores to each bot depending on whether the bot answered appropriately. The scores on each question were summed to generate an overall knowledge score. To assess specific knowledge, Kuligowska et. al. asked a series of domain-specific questions that could be relevant across domains (e.g. What product are you selling? What is the price of your product?) [102]

Fifth, the researchers assessed the chatbots' ability to present their knowledge along five dimensions: (i) did the bots open click-through links in a new tab or window so as not to disrupt the dialogue, (ii) did the bots provide an "interactive connection to an external knowledge database," (iii) does the bot provide a back button or the ability to scroll through past dialogue, (iv) are chatbots able to provide explanations for their functioning after receiving inputs such as "?", "info", and "help," and (v) do chatbots have a Home button that links back to the Main Menu. [102]

Sixth, Kuligowska et. al. evaluate conversational abilities. Chatbots that are able to lead a coherent dialogue, understand context, and repair 'broken' conversations receive the highest 5/5 ratings. Those bots that may not be able to fully understand context but that can lead a coherent dialogue receive a rating of 4/5. Finally, those bots that can only understand a small number of utterances, and will frequently respond with the same blanket phrases, receive 3/5 ratings. [102]

Seventh, Kuligowska et. al. assess whether chatbots have consistent and rich personalities, based on the assumption that for chatbots to act like believable humans, they must be able to simulate having a unique personality. Those bots such as Orange's Ewa with rich personalities received ratings of 5/5, while those chatbots with only a "rudimentary outline of personality" received ratings of 3/5. [102]

Eighth, Kuligowska et. al. evaluate the depth of personalization options for the chatbot, along five dimensions: (i) is the user empowered to select the gender of the bot, (ii) can the bot recall the name of the user, (iii) can the user view the conversation history during the talk, (iv) can the bot send the conversation history via email or print to the user, and (v) can the bot recognize the sub-page of the website that the user is browsing. [102]

3. Architecture & Design

3.1. Speech-to-Text Conversion

3.1.1. Large Vocabulary Speech Recognition

Speech is one the most natural and powerful modes of communication, and is "widely accepted as the future of interaction with computer and mobile applications." [13] Speech is so integral to the human condition that people with IQs of less than 50, and with brains one third the size of humans can speak. Neurological research indicates that speech activates more of the brain than any other processing function. By incorporating speech processing, chatbots will be able to interface over phones and radios. [14]

Speech-to-text conversion begins with a process called automatic speech recognition (ASR). The goal of ASR is to achieve speaker-independent large vocabulary speech recognition (LVCSR). [15] Improvements in LVCSR can be measured along a number of dimensions:

A. **Vocabulary size**: vocabularies started out miniscule, and only included basic phrases (e.g. yes, no, digits, etc.) and now include millions of words in many languages. [15]

B. **Speaker independence:** ability to recognize specific speakers. This is only relevant if chatbots use the speaker's identity to generate user-specific responses. [15]

C. **Co-articulation:** ability to process a continuous stream of words, which do not necessarily contain breaks between words. Requires proper tokenization and segmentation of the input stream, discussed in the next section. [15]

D. **Noise handling:** ability to filter out noise (e.g. traffic, background music or speech). [15]

E. **Microphone:** ability to process speech at varying distances from microphone. [15]

There is not a direct one-to-one matching between a sound and a corresponding phoneme or word: each time a speaker makes attempts to communicate a word, the resulting sound will be different because of: (i) environmental noise, (ii) emotional state, (iii) tiredness, and (iv) distance from microphone, among other factors. The matching process in ASR is therefore not deterministic, but rather can be modeled as a stochastic process. Given a sound X, the model generates the most likely phoneme, word (W), phrase, or sentence from all possible words in the Language L. [15]

3.1.2. ASR Process Model

The first step in ASR is pre-processing. Speech is recorded into a microphone; Apple's Siri, for example, uses iPhone hardware, whereas hardware (including microphone) for Amazon's Alexa must be purchased independently of a mobile phone. The signal is discretized by the microphone with a sampling frequency (16kHz is empirically optimal for speech). [15]

The second-step is speech/non-speech segmentation. The ASR system must distinguish between the phonemes (basic unit of speech) that should be recorded for translation vs. the background noise, which can also be recorded as phonemes but is not relevant to translation. Likewise, the system should remove phonemes before the actual speech began, and after the desired recording took place; this is called end point detection and removes noise. Signal energy-based algorithms, which set energy thresholds that are crossed when speech begins and ends, as well as Gaussian mixture models can be used to solve this problem. [15]

The third step is feature extraction. Acoustic observations are "extracted over time frames of uniform length," typically 25 milliseconds. The fourth step is decoding, in which the acoustic feature vectors are mapped to the most likely corresponding words. This requires three tools. [15]

First, we need an acoustic model which will give us P(X|W) – given a word, the probability that we hear a sound. We will find the argmax over all words in our language w, or the word with the highest likelihood of representing this sound. Acoustic models are typically trained on sound recordings and accompanying transcripts, which can be used to empirically find these probabilities. The statistical representation of each word (or phoneme) generated from analysis of the sound corpus is typically represented as a Hidden Markov Model (HMM). HMMs are Markov process where the states are unobserved/hidden (we do not know the actual phonemes/words used, and are approximating this information). [15]

Second, we need a language model which can tell us the probability of hearing a given word in our language. Third, we need a dictionary with a list of words and their phonemes. [15] Given a sound, we can decode the word using Bayes rule:

$$W = argmax_{W\in L} P(W|X) = argmax_{W\in L} \frac{P(X|W)P(W)}{P(X)} = argmax_{W\in L}\, P(X|W)P(W)$$ [16]

The fourth step is post-processing; Bayes rule gives us a list of the probable word sequences with their ranks, and we choose the most probable one as the word sequence we've heard. The other top hypotheses are stored and can be used in reinforcement learning algorithms later, where they will be used to learn from and correct mistakes in the ASR phase. [15]

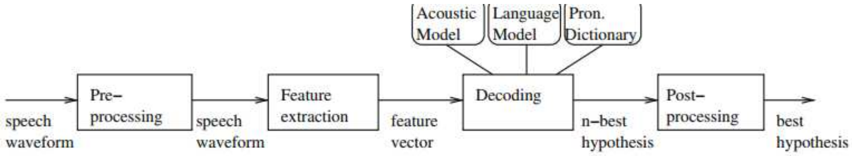


Figure 1: Automatic Speech Recognition (ASR) Process [15]

3.1.3. Restricted Boltzmann Machine (RBM) Implementation

Deep learning and the use of Restricted Boltzmann Machines (RBMs) have led to the development of even more effective methods for decoding sounds into phonemes. RBMs are neural networks with one layer of stochastic visible units and N layers of stochastic hidden units; there are no connections with each layer, but there typically connections between each unit in the visible layer and every unit in the hidden layer. The weights on each edge are determined by an activation function, and are altered and optimized during training via back propagation. [17]

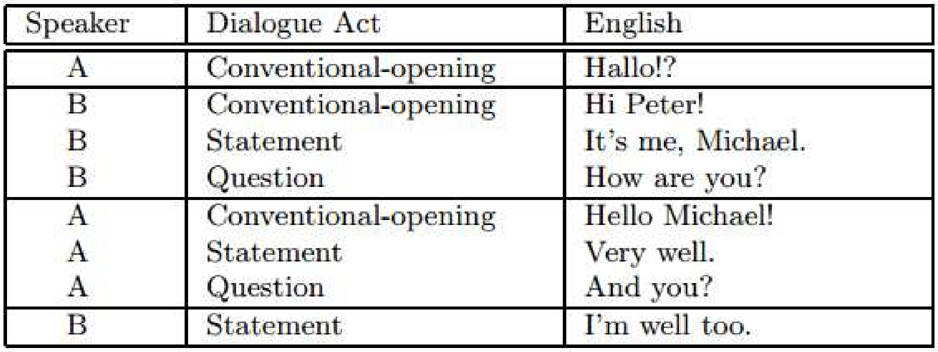
Rather than include an HMM to represent the sequence of sounds that make up a phoneme, RBM-based models represent each phoneme with an RBM, where each sound frame that composes the phoneme is a visible unit. Mohamed et. al. tested three variants of the RBMs in acoustic modeling: the unconditional RBM, a conditional RMB (CRBM), which considers the visible variable from previous time frames as conditional inputs, and the interpolating CRBM (ICRBM), which considers both previous and future time frames as conditional inputs. Of these, they find the ICRBM generates the best results, and outperformed standard feed-forward neutral nets. [17]

3.2. Natural Language Processing

The goal of natural language processing (NLP) is to take the unstructured output of the ASR and produce a structured representation of the text that contains spoken language understanding (SLU) or, in the case of text input, natural language understanding (NLU). In this section, we explore a number of methods for extracting semantic information and meaning from spoken and written language in order to create grammatical data structures that can be processed by the Dialogue Management unit in the next step. This is non-trivial because speech may contain: (i) identity-specific encodings (e.g. pitch, tone, etc.) in addition to meaning-encodings and (ii) noise from the environment. Likewise, both speech and text inputs to a chatbot may contain (iii) grammatical mistakes, (iv) disfluencies, (v) interruptions, and (vi) self-corrections. [18]

3.2.1. Dialogue Act (DA) Recognition

One way to extract meaning from natural language is to determine the function of the text/sentence (e.g. is this a question, suggestion, offer, or command); this is called dialogue act recognition. In dialogue act recognition systems, a corpus of sentences (training data) is labeled with the function of the sentence, and a statistical machine learning model is built which takes in a sentence and outputs its function. The model uses a number of different features to classify the sentences including: (i) words and phrases such as “please” (function=request) and “are you” (function=yes/no question), and (ii) syntactic and semantic information. [16]



Speaker	Dialogue Act	English
A	Conventional-opening	Hallo!?
B	Conventional-opening	Hi Peter!
B	Statement	It’s me, Michael.
B	Question	How are you?
A	Conventional-opening	Hello Michael!
A	Statement	Very well.
A	Question	And you?
B	Statement	I’m well too.

Figure 2: Dialogue Act Recognition Example [19]

The first-step in creating a dialogue act recognition system, is defining the relevant functions or the DA tag-set. This involves choosing labels that are general enough to be re-used in multiple tasks, specific enough to remain relevant for the target task, and clear/separable enough that there is little confusion for humans in labeling the functions of sentences in the training set. A number of tag-sets have gained prominence and are the most frequently used in chatbots: Dialogue Act Markup in Several Layers (DAMSL), Switchboard SWBD-DAMSL, Meeting Recorder, VERBMOBIL, and Map-Task. [19]

The DAMSL annotation scheme labels a sentence in four dimensions: communicative status, information level, forward-looking function, and backward-looking function. Communicative status labels a sentence as: (i) uninterpretable, (ii) abandoned, or (iii) self-talk. Information level labels a sentence as: (i) task, (ii) task management, (iii) communication-management, or other. Forward-looking functions encode any information that will affect future conversation and label sentences along eight sub-dimensions: (i) statement – assert, reassert, or other, (ii) influencing-addressee-future-action – open-option or action directive, (iii) info-request, (iv) committing-speaker-future-action – offer or commitment, (v) conventional opening-closing, (vi) explicitperformative, (vii) exclamation, or (viii) other. Backward-looking functions encode the relationship between current and previous speech, such as (i) agreement, (ii) understanding, (iii) answer, or (iv) information relation. [20]

Switchboard is an adaptation of DAMSL for automated telephone conversations. [21] Meeting Recorder DA (MRDA) is similar to Switchboard: its training Corpus, however, is labeled recordings of 72 hours of conversations from meetings. MRDA handles intricacies well given the complications that often occur during meetings such as speaker overlap, frequency of abandoned comments, and complicated turn-taking interactions. [22] Finally, Map Task is a hierarchy of levels: the first level labels transactions, the second is conversational games, which labels patterns like question and answer pairs, and the third includes 19 conversational moves. [23]

3.2.2. Bayesian Approaches to DA Models

The idea behind using a Bayesian approach to DA models is to find the probability of every possible sequence of dialogue acts DA that could represent a sentence or utterance (U), and find the dialogue act sequence DA_{Max} with the highest probability of occurring. [16]

$$DA_{Max} = argmax_{DA}\, P(DA|U) = argmax_{C}\, \frac{P(DA)P(U|DA)}{P(U)} \text{ [16]}$$

$$DA_{Max} = argmax_{DA}\, P(DA) * P(U|DA) \text{ [16]}$$

Assuming the N individual words of the utterance are known, and the Naïve Bayes assumptions of independence of subsequent words holds, this leads to:

$$DA_{Max} = argmax_{DA}\, P(DA) * \prod_{i=1..N} P(W_i|DA) \text{ [16]}$$

This is the unigram model, where P(DA) and $P(W_i|DA)$ can be quantified from empirical data. Using this Naïve Bayes classifier, Reithinger et. al. finds a 74% recognition/accuracy rate for classifying the correct dialogue act from a given sentence. [24] N-gram models are frequently used to include dialogue history in the model. These models estimate P (DA | N historical DAs) rather than P(DA). Assuming N = 3, we would estimate $P(DA \mid DA_{n-1}, DA_{n-2}, DA_{n-3})$. [25] Hidden Markov Models can also be used to model dialogue history, where each state represents a dialogue act in the conversation history of the chatbot. [26] Likewise, neural network classifiers can be trained as well. A combination of HMMs and neural networks has achieved 76% accuracy. [27]

3.2.3. Non-Bayesian Approaches to DA Models

DA-classification is a classic machine learning problem. A number of non-NB approaches have been used, including: neural networks, multi-layer perceptrons, and decision trees. [19]

Wright, 2015, for example, implemented a multi-layer perceptron. The input layer to the MLP neural network used suprasegmental features (e.g. stress and intonation) extracted from the utterance, durational features (i.e. time taken to voice word), and prosodic features (e.g. root mean square frame energy). The neural network used one hidden layer with 60 neurons, and an output layers with 12 nodes, corresponding to each of the 12 possible DA configurations/labels being used. [28] The network was trained using back-propagation with a cross-entropy cost function:

$$DA = -\ln \sum_x [yln(a) + (1-y)\ln(1-a)]$$ [29]

N is the number of training data items, x is the training inputs/features, and y is the outputs in each round of back-propagation. [29]

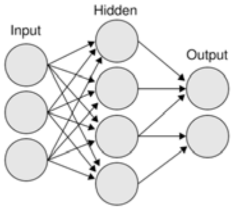


Figure 3: MLP Example [19]

Andernach et. al., by contrast, used a different type of neural network: the Kohonen network or Self-Organizing Map (SOM), which maps from a set of data (the input features) onto a two-dimensional output grid representing the possible DAs. Seven input features were used including: speaker, use of "wh" question word, use of question mark, and other features. These networks use unsupervised clustering to create the labels used to classify input sentences. [30]

A third non-NB approach to DA classification is memory-based learning (MBL) using K-nearest-neighbor. The idea behind MBL is to store in memory all instances that have been seen and classified (i.e. at first, the training data). When an unseen instance (new DA) is seen, the system retrieves the k-nearest-neighbors (i.e. the most similar utterances from the training set) and the unseen instance is classified based on the majority or "dominant" class in this sample. [19] Rotaru achieves optimal accuracy of 72% using the 3-nearest-neighbor approach. [31] A fourth approach is transformation-based learning in which a simple classifier is used to get a generally-correct solution, and transformations are applied to get a specifically-correct solution. [16]

DA classification is critical to chatbots in that it provides necessary context to the bot and directs future communication. This, however, is somewhat of a circular problem in that DA classifiers would be much improved if they themselves had context information. DA classifier development is focused on providing more relevant context information, including social roles of users, relationships, emotions, context, and history of interaction as classification features. [19]

3.2.4. Intent Identification

In SLU, the functions / dialogue acts (DAs) are often domain specific. In other words, instead of asking whether the function of the user's utterance is a question or answer, we ask whether the function is to, for example, find flights or cancel a reservation in a flight reservation program. [32]

Domain-specific dialogue acts are called intents. Intent identifying has been most prominently used by call center bots, which ask the user "how can I help you?" and subsequently use intent identification to re-direct the user to one of N pre-defined re-direction options. [18] Many of the same machine learning algorithms used for DA classification are used for intent identification. [16]

3.2.5. Information Extraction

The primary responsibility of the SLU is not just to understand phrase function, but to understand the meaning of the text itself. To extract meaning from text, we convert unstructured text – either the output of the ASR, or text written into a text-only chatbot – into structured grammatical data objects, which will be further processed by the Dialogue Manager.

The first step in this process is breaking down a sentence into tokens that represent each of its component parts: words, punctuation marks, numbers, etc. Tokenization is difficult because of the frequency of ambiguous or mal-formed inputs including: (i) phrases (e.g. New York), (ii) contractions (e.g. aren't), abbreviations (e.g. Dr.), and periods (e.g. distinguishing those used in "Mr." and at the end of a sentence). These tokens can be analyzed using a number of techniques, described below, to create a number of different data structures that be processed by the dialogue manager. [16]

Bag of Words: We ignore sentence structure, order, and syntax, and count the number of occurrences of each word. We use this to form a vector space model, in which stop words (e.g. a, the, etc.) are removed, and morphological variants (e.g. talk, talks, talked, etc.) go through a process call lemmatization and are stored as instances of the basic lemma (e.g. talk). In the dialogue manager phase, assuming a rule-based bot, these resulting words will be matched against documents stored in the bot's knowledge database to find the documents with inputs containing similar keywords. The bag of words approach is simple because it does not require knowledge of syntax, but, for this same reason, is not precise enough to solve more complex problems. [16]

Latent Semantic Analysis (LSA): This approach is similar to the bag of words. Meanings / concepts, however, not words, are the basic unit of comparison parsed from a given sentence or utterance. Second, groups of words that co-occur frequently are grouped together. In LSA, we create a matrix where each row represents a unique word, each column represents a document, and the value of each cell is the frequency of the word in the document. We compute the distance between the vector representing each utterance and document, using singular value decomposition to reduce the dimensionality of the matrix, and determine the closest document. [16]

Regular Expressions: Sentences / utterances can be treated as regular expressions, and can be pattern matched against the documents in the bot's knowledge database. For example, imagine that one of the documents in the bot's knowledge database handles the case where the user inputs the phrase: "my name is *". "*" is the wildcard character, and indicates that this regular expression should be triggered whenever the bot hears the phrase "my name is" followed by anything. If the user says "my name is Jack", this phrase will be parsed into a number of regular expressions, including "my name is *" and will trigger the retrieval of that document. [16]

Part of Speech (POS) Tagging: POS tagging labels each word in the input string with its part of speech (e.g. noun, verb, adjective, etc.). These labels can be rule-based (a manually-created set of rules is created to specify part of speech for ambiguous words given their context). They can also be created using stochastic models which train on sentences labeled with correct POS. In the dialogue manager, POS can be used to store relevant information in the dialogue history. POS is also used in response generation to indicate the POS object type of the desired response. [16]

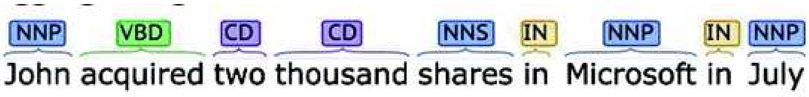


Figure 4: Sentence labeled with POS [16]

Named/Relation Entity Recognition: In named entity recognition (NER), the names of people, places, groups, and locations are extracted and labeled accordingly. NER-name pairs can be stored by the dialogue manager in the dialogue history to keep track of the context of the bot's conversation. Relation extraction goes one step further to identity relations (e.g. "who did what to whom") and label each word in these phrases. [16]

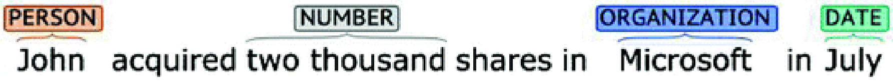


Figure 5: Sentence labeled with Information Extraction [16]

Semantic Role Labeling: The arguments of a verb are labeled based on their semantic role (e.g. subject, theme, etc.). In this process, the predicate is labeled first followed by its arguments. Prominent classifiers for semantic role labeling have been trained on FrameNet and PropBank, databases with sentences already labeled with their semantic roles. These semantic role-word pairs can be stored by the dialogue manager in the dialogue history to keep track of context. [16]

Creation of Grammatical Data Structures: Sentences and utterances can be stored in a structured way in grammar formalisms such as context-free grammars (CFGs) and dependency grammars (DGs). Context-free grammars are tree-like data structures that represent sentences as containing noun phrases and verb phrases, each of which contain nouns, verbs, subjects, and other grammatical constructs. Dependency grammars, by contrast, focus on the relationships between words. [16]

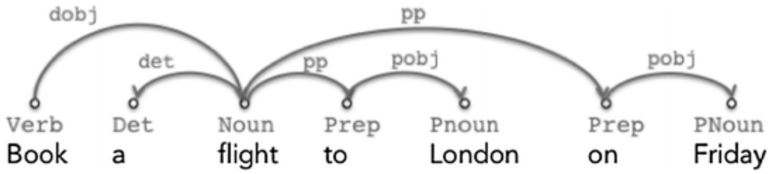


Figure 6: Example of Dependency Grammar parsing [16]

3.2.6. Statistical Methods for Information Extraction

Historically, the above hand-crafted models, created based on knowledge of the specific situation at hand, were used to extract structured meaning from a sentence or utterance. [16]

However, these models have two problems. First, hand-crafted models lead to high development costs, because new models must be built with each new system. Second, these models are often not robust to diverse user input, because users do not know which grammatical structures are handled by a specific system. To solve these problems, data-driven, statistical models have arisen. These models derive the meaning M of an utterance U that maximizes P(M|U). [16]

Hidden Vector State (HVS) Model: He and Young offer a statistical hidden vector state model. Given a sentence, our goal is to automatically produce some accurate structured meaning. Consider the sentence "I want to return to Dallas on Thursday." The parse tree below represents one way of representing the structured meaning of the sentence. SS represents the initial node send_start, and SE represents the end node send_end. We view each leaf node as a vector state, described by its parent nodes: the vector state of Dallas is [CITY, TOLOC, RETURN, SS]. [33]

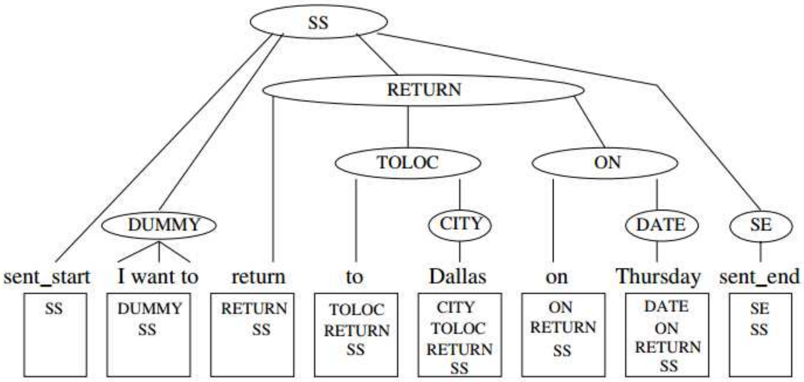


Figure 7: Parse tree of HVS [33]

The whole parse-tree can then be thought of a sequence of vector states, represented by the sequence of squares above. If each vector state is thought of as a hidden variable, then the sequence of vector states (e.g. squares above) can be thought of as a Hidden Markov Model: we start at SS, and have certain probabilities of reaching a number of possible hidden states as the next state. Each vector state can be thought of as a "push-down automaton" or stack. This way, the first vector state is SS and after that, our transitions are a combination of push or pull elements to get to the next vector state. At each step, we can make up to n stack shifts as our transitions. If we do not restrict n, the state space will grow exponentially (e.g. n possible states from S, n^2 possible states in the next step, etc.). To prevent this from happening, we limit the maximum depth of the stack. [33]

With this model in mind, He and Young argue that the probability of a set of words W occurring with a corresponding set of concepts C, and set of transition operations N is the product of the (i) the probability of the t-th stack operation n_t occurring given the last concept c_{t-1}, (ii) the probability of the t-th concept from the root node $c_t[1]$ occurring given the sequence of previous concepts $c_t[2..D_t]$ and (iii) the probability of the t-th word w_t occurring given the t-th concept c_t, for all t=1…T, where T is the final state of the model: [33]

$$P(N,C,W) = \prod_{t=1 \ldots T} P(n_t|\, c_{t-1})P(c_t[1]|c_t[2..D_t])P(w_t|c_t\,)\ [33]$$

To train the model, we create a set of pairs of sentences and their structured concept annotations, and fit model parameter λ that allows us to convert from sentences to concepts. In training, we seek to maximize:

$$\mathrm{L}(\lambda) = \log P(N, C, W \mid \lambda) \text{ [33]}$$

When we run the model on unseen sentences (e.g. in prediction / classification), we apply the model parameters derived in training to generate our desired output: the appropriate structured concept annotations. [33]

Other similar models include: the stochastic finite state transducer model, which uses finite state machines to determine the appropriate grammatical concept with which to label a given word, and dynamic Bayesian network models. All of these models are generative models in that they seek to find the joint probability distribution P(X,Y) where X is the sentence inputs and Y is the structured grammatical concept outputs. Next, we discuss discriminative which calculate the conditional probability of P(Y|X) in order to map sentences to concepts. [16]

Support Vector Machine (SVM) Model: Support Vector Machines are a supervised machine learning tool. Given a set of labeled training data, the algorithm generates the optimal hyperplane that divides the sample into their proper labels. Traditionally, SVMs are thought of as solving binary classification problems, however multiple hyperplanes can be used to divide the data into more than two label categories. The optimal hyperplane is defined as the hyperplane that creates the maximum margin, or distance, between different-labeled data point sets. [34]

Traditionally, a hyperplane is represented as H(X) = W*X + B = 0, where X is the input points, W is a weight vector, and B is the bias term. SVMs are used to achieve SLU by treating SLU as a sequence of binary classification problems mapping words to concepts. [16]

Conditional Random Field Models: CRFs are log-linear statistical models often applied for structured prediction. Unlike the average classifier, which predicts a label for a single object and ignores context, CRF's take into account previous features of the input sequence through the use of conditional probabilities. A number of different features can be used to train the model, including lexical information, prefixes and suffixes, capitalization and other features. [16]

Deep Learning: The most recent advancement in the use of statistical models for concept/ structure prediction is deep learning for natural language processing, or deep NLP. Deep learning neural network architectures differ from traditional neural networks in that they use more hidden layers, with each layer handling increasingly complex features. As a result, the networks can learn from patterns and unlabeled data, and deep learning can be used for unsupervised learning. Deep learning methods have been used to generate POS tags of sentences (chunk text into noun phrases, verb phrases, etc.) and for named-entity recognition and semantic role labeling. [16]

3.3. Response Generation

Response generation is arguably the most central component of the chatbot architecture. As input, the Response Generator (RG) receives a structured representation of the spoken text. This conveys information about who is speaking, the dialogue history, and the context. As output, the RG generates a response to deliver to the user, which it will deliver to the Dialogue Manager (DM).

The response selector has access to three key components it will use to make its decision about what to say: (i) a knowledge database / data corpus, which will differ in content based on implementation, (ii) a dialogue history corpus, which will only exist in more complex models, and (iii) an external data source, which provides the bot with intelligence (e.g. a dog is an animal). This latter "common sense intelligence" is often achieved by allowing bots to access and retrieve documents from search engines.

In this section, we will discuss in detail the ways in which a bot can retrieve a response.

3.3.1. Rule-Based Models

The principle idea underlying rule-based RG is that a bot contains a knowledge base with documents, where each document contains a <pattern> and <template>. When the bot receives an input that matches the <pattern>, it sends the message stored in the <template> as a response. The <pattern> could either be a phrase "What's your name?" or a pattern "My name is *", where star is a regular expression. Typically, these <pattern> <template> pairs are hand-crafted. [6]

The programmer's main challenge is to choose an algorithm to use for pattern matching between the input sentence and the documents in the data corpus. We can think of this as a nearest-neighbor problem, where our goal is to define the distance function, and retrieve the closest document to the input sentence. Below, we include some of the most common methods.

ELIZA Incremental Parsing: ELIZA, an early chatbot created at MIT between 1964 and 1966, used an incremental parsing or "direct match" approach for pattern matching. ELIZA parsed the input text word by word from left to right, looking each word up in the dictionary, giving it a rank based of importance, and storing it on a keyword stack. The keyword with the highest rank of importance would be treated as the <pattern> and pattern matched against the documents in the corpus to find the appropriate template response. If no document existed corresponding to the input, ELIZA would say "I see" or "Please go on" in the DOCTOR script. ELIZA was created with multiple "scripts" which indicate different <template> responses to <pattern> inputs. [35]

ALICE/AIML Direct Match Technique: ALICE-bot is a later chatbot implementation and three-time winner of the Loebner prize. The ALICE architecture includes a knowledge base called Graphmaster, which is a graph with nodes ("nodemappers") and edges. Graphmaster can be thought of like a file system, with a root that contains files and directories. The name of the path to every leaf node is that leaf node's <pattern> sentence. We conduct pattern matching using depth-first search on the knowledge-base. Let's assume our sentence starts with the word "Hello" and we start in Folder. Start by checking for a sub-folder that starts with "_". Scan through the sub-folder to look for matches with any remaining suffixes from the input sentence. If no match is found, return to the Folder and look for the sub-folder Hello, and scan through the sub-folder to look for matches with any remaining suffixes, minus the word Hello. If no match is found, return to the Folder and look for the sub-folder *. Scan through the sub-folder to look for matches with any remaining suffixes, minus the word Hello. If no match is found, retrieve the second word and repeat this process. Recursively continue until a match is found or you finish the sentence. [36]

VPBot Keyword Set Matching Technique: Incremental parsing and direct match both require only one word to match in order to retrieve a response (e.g. one keyword match in the former, and one direct match in the latter). This flexibility could lead to unexpected behavior. VPBot modified this to technique by allowing developers to assign between 1-3 keywords to a keyword set, and required that all keywords in the set be present to trigger a response. [35]

ViDi One-Match and All-Match Categories (OMAMC) Technique: OMAMC combines the ideas of the ALICE and VPBot matching techniques by creating two matching mechanisms. The first mechanism is one-match, which is an exact-match technique. If there is an exact match between keywords (can be single word or phrase) in the input sentence and a document's one-match category in the knowledge base, this document will be retrieved. The second mechanism is all-match, which allows the programmer to create keyword sets that are larger than three (the limit set by VPBot). If all keywords in a document's all-match category appear in the input sentence, regardless of the ordering, this document is retrieved. If two different documents are retrieved via one-match and all-match, one-match has precedence (in practice, all-match is not run unless one-match does not retrieve a document). OMAMC has two benefits. First, it allows for varied keywords arrangements in documents (order does not matter in all-match). Second, it allows for keyword variety (no limit of three, like in VPBot, in all-match). [35]

3.3.2. Information Retrieval (IR)-Based Models

The main problem with rule-based systems is that the programmer must specify all "rules" or <pattern> <template> pairs. With the advent of large datasets such as dialogues on Reddit and Twitter, this no longer became necessary. Instead, developers could analyze millions of conversations and store these conversations in documents as pairs of <statuses> (first speaker) and <responses>. The principle behind retrieval-based systems is, given an input sentence, to pattern match this against the set of <status> <response> pairs and select a response.

The main challenge in Information Retrieval (IR) algorithms is determining how to conduct pattern matching. First and foremost, there is the consideration of, given an input sentence, whether to pattern match against the <status> or <response>. The former seems more intuitive, and would involve finding the most similar <status> in the data corpus. The latter, however, is used more often in practice because it is more effective, and involves comparing the input sentence against the <responses> in the data corpus. The reason matching against <responses> is more effective is that if words co-occur between the input sentence and a <response>, it is likely an appropriate response. By contrast, just because words co-occur in the input sentence and <status>, if it is not an exact match, we don't know if the <response> will be appropriate. Likewise, if the system pattern matches against responses, not inputs, it can simply use search engines to query for a set of responses that are similar to the input phrase (e.g. search engines retrieve response sets). [37]

Second, regardless of whether we are pattern matching against <status> or <response>, we must choose a pattern matching algorithm. Here, we could use a basic keyword approach, in which we look for keywords that co-occur in the input sentence and document, and this is occasionally used. More complex IR models, however, typically use an ensemble of score functions, calculate the overall rank of each document as a weighted average of those scores, and return the document with the highest score. [37]

Given a status (S) – the input sentence – and a corpus of documents (D) in the knowledge base, these algorithms return:

$$Rank(S,D) = \sum_{k} \lambda_k * score_k(S,D)$$ [37]

In this formula, the output of the k-th score function $score_k(S,D)$ is multiplied by the weight received by that score function λ_k. The sum across all score functions gives the overall rank. [37]

Yan et. al. use a number of score functions. First, they count the number of common words in S and D, excluding stop-words (“a”, “the”, etc.), weighted by inverse document frequency (IDF). IDF values indicate how common a word is in the data corpus. For example, octopus typically receives a higher IDF value than school, because the term is used less frequently in most corpuses of text. By weighting by IDF score, we raise the score for documents with rare words that co-occur with the input sentence. Second, they measure the average cosine distance between the word embeddings in the input sentence and words in the document, excluding stop words. [37] Word embeddings are mappings of words to mathematical vectors. [38]

A weighted average of these and other scores gives each document a rank, and the highest ranked document is returned. Other more complex score functions used by Yan et. al. include statistical machine learning phrase tables and convolution neural networks. [37]

Use of neural networks to rank response choices is quite common. Wu. et. al.’s chatbot response selection model, for example, uses neural networks to produce a score for each potential document. Their model takes as inputs an utterance u and a response candidate r. At the first layer of the model, we break the utterance and response down into word embeddings.

$U = e_{u,1}, \dots, e_{u,n_u}$ and $R = e_{r,1}, \dots, e_{r,n_r}$ [39]

From these work embeddings, we construct a word-word similarity matrix M_1 and a sequence-sequence similarity matrix M_2. Constructing $M_1$is relatively simple: we set every element:

$e_{i,j} \in M_1 = e_{u,i}^{\perp} * e_{r,j}$. [39]

Constructing M_2 is significantly more complex, and requires the use of a recurrent neural network with gated units (GRU). The GRU models “the sequential relationship and dependency among words” in a sequence of words from word i=1 up until word i=r (in response) and word i=u (in utterance). [39] Chung et. al. provides a full description of the GRU’s functionality. [40] Simply put, the GRU allows the model to track relationships between the input sentence and response candidate on a segment or phrase level, rather than simply on a word by word level. [39]

These matrices are input channels to a convolution neural network (CNN), which outputs a vector v that encodes the similarities between u and r based on the data from $M_1$and M_2. The CNN learns from training data which phrases in a given input utterance, and subsequently which GRU outputs, are useful in identifying responses with high similarity. During prediction, these rules will be applied to select specific features of the input channels by convolution and pooling, move these into the output vector, generate a score for each response, and choose the best one. [39]

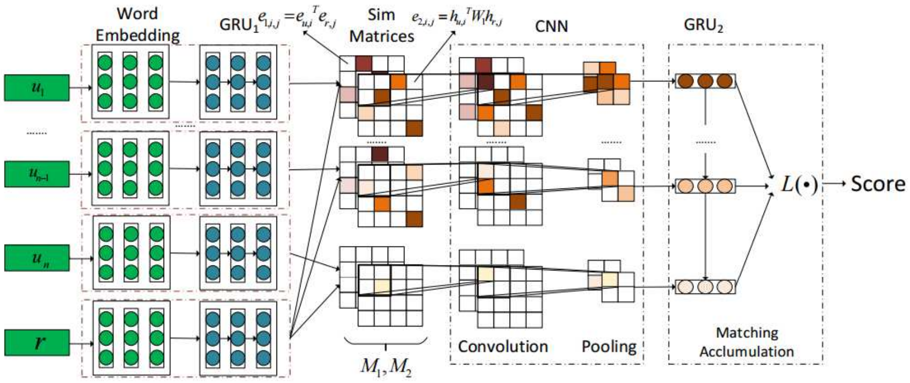


Figure 8: RNN used in IR [39]

This approach is too complex to be run on every response candidate. Wu et. al. use a basic rule-based pattern-matching approach to select the top five response candidates. These responses are run through the RNN above and re-ranked to select the top response. [39]

3.3.3. Statistical Machine Translation Generative Models

Information retrieval models require a database of possible responses to choose from. Generative models, by contrast, build responses on the fly. Most do so using machine learning techniques: classifiers are trained based on real dialogues, and used to generate responses to users by "translating" inputs into responses. Statistical Machine Translation (SMT) models are some the most recent, and effective, models used to generate chatbot responses.

Statistical Machine Translation is a field of machine translation which analyzes bilingual corpuses of text, and uses statistical analysis to derive exact transactions between individual words, phrases, and features of the texts. The benefit of machine translation over human translation is first and foremost resource savings. Moreover, SMT has access to many parallel corpora of languages allowing for analyses to generate translations between multiple languages. SMT generates cost savings relative to other automated approaches in that rule-based translation systems require the manual creation of rules. This is costly, and creates rules that typically only work for one language pair because they do not generalize well for other language translations. [41]

Ritter et. al. proposed in 2011 the first use of SMT in chatbot response-generation. They proposed the creation of a machine translation system that could translate between an input in one language, and a response to that input in the same language based on corpora of Twitter dialogue. Historically chat-bots have been limited to specific domains (e.g. ELIZA is a Rogerian psychotherapist, and cannot answer questions about cooking). Ritter's chatbot is restricted to the Social Media domain: in other words, because social media covers nearly every conceivable topic, the chatbot has extensive knowledge. The second benefit of Ritter's proposed chatbot is while historically chatbot used canned responses and templates (e.g. all retrieval-based systems), a generative bot model creates utterances as a direct function of specific input words, and therefore has the potential to generate more natural responses. [42]

SMT models applied to conversation generation take the previous user's comment c as input, and generate a response r using a log-linear model, based on a number of features. Features include: P(C|R), the probability of given comment C given the response R, P(R|C), the probability of a response R given a comment C, the language model P(R), which ensures that R is a reasonable phrase that could be generated in conversation. The language model is build using n-gram statistics (we count the appearance of n-grams in our training data, and estimate if the n-grams that appear in R exist in or are well-predicted by the training data). During training, a translation table is also created based on the training data (e.g. Twitter dialogue conversations). [42]

In prediction, Ritter et. al.'s model uses the Moses phrase-based decoder to choose the best response given the input sentence. [43] The decoder goes left to right and translates phrases from the input sentence, generating a number of possible hypotheses at each step. Without pruning the number of hypotheses would grow exponentially with the length of the input sentence because all hypotheses for the translation phrase n+1 are added to the hypotheses for phrase n, and so on. [44]

To prevent this from happening, hypotheses that are deemed similar are combined throughout this process. Second, pruning occurs. As words are translated, and these hypotheses are added to previous hypotheses sets, we calculate the cost (i.e. weakness) of each set of hypotheses and estimate their future cost. Current translation costs are estimated using the language model which accounts for the probability of these words occurring in the target language (in our case, the response set). Current costs also include distortion costs which account for the likelihood of combinations of phrases co-occurring. The highest cost hypotheses are discarded until we reach the end of the input sentence, and choose the lowest cost option as our selected output. This prediction process is called beam search. [44]

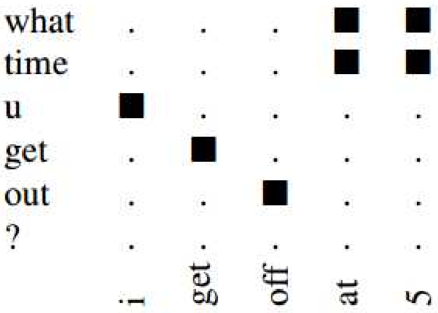
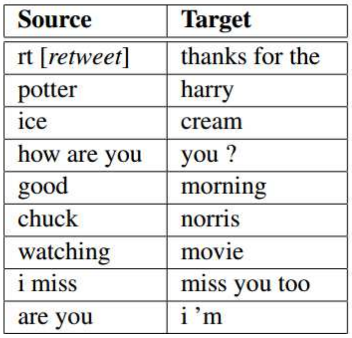


Figure 10: Example output of SMT Model [42]



Source	Target
rt [*retweet*]	thanks for the
potter	harry
ice	cream
how are you	you ?
good	morning
chuck	norris
watching	movie
i miss	miss you too
are you	i 'm

Figure 9: Top input-output phrase pairs [42]

Challenge of Lexical Repetition: When SMT is applied to response generation, a problem occurs: given an input sentence, the model generates the same, or a very similar, sentence as the response. This occurs because in dialogue, the responder typically uses the same language as the initiator. To solve this challenge, Ritter et. al. filters out any translations where the translation is a substring of the initial language and penalizes responses that are similar to input phrases by calculating the Jacard similarity between the input and response. [42] The Jacard similarity index is typically calculated as a distance score between the bag of words in the input and response. [45]

Challenge of Word Alignment: The challenge in word alignment involves long input sentences in which there in not a 1-1 correspondence between words/phrases in the input sentence and words/phrases in the output sentence. [42] Ritter et. al. use Giza++, the refined alignment model created by Och and Ney, a widely used tool which empirically outperformed most existing alignment tools, to pair phrases in input sentences and responses. [46] Still, they achieve an alignment score of 1.7 compared with the standard score of 14 for translation between French and English phrase pairs, indicating that these sentences may not align well. This problem has not been sufficiently solved. As a result, Ritter et. al. uses Fisher's Exact Test, a test of statistical significance, to remove pairs of input-output phrases that do not align. [42]

Despite these challenges, SMT achieve relatively strong results: in a Mechanical Turk study, 65% of crowd-workers preferred SMT to Information Retrieval where input is compared against document <status> and 59% preferred SMT to Information Retrieval where input is compared against document <response>. SMT was preferred over human response 15% of the time. [42]

3.3.4. Sequence to Sequence (Seq2Seq) Model

Cho et. al. in 2014 proposed a variation of Ritter's generative model that uses advances in deep learning to achieve higher accuracy. Their Encoder-Decoder model uses two recurrent neural networks (RNN). The first, encodes the <status>, or input sentence, and the second decodes the <status> and generates the desired <response>. In doing so, the model is able to translate between language (e.g. for statistical machine translation) and can also be used to enable chatbots to convert between document <status> and document <response>. The Sequence to Sequence model (Seq2Seq) is the current industry best practice for response generation, and is widely used. [46]

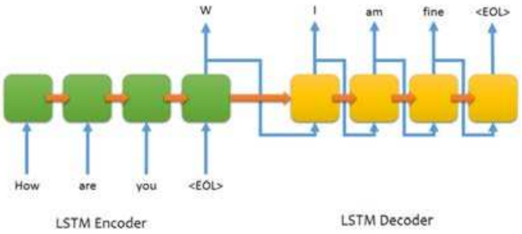


Figure 11: Example of Seq2Seq RNN [47]

Recurrent neural networks (RNN) are neural networks with a hidden state h and output y, which is a function of a sequence $X_1 \dots X_n$. In our case, the sequence is the input string to the chatbot. At each time step, the RNN's hidden state is updated by the below function. [46]

$$h_t = f\ (h_{t-1,}x_t)\ [46]$$

F is an activation function such as the sigmoid function or long short-term memory (LSTM) unit. The goal of the encoder RNN in Cho et. al.'s model is to predict the following function. [46]

$$P\left(y_1 \dots y_{ty} \middle| x_1 \dots x_{tx}\right)\ [46]$$

The lengths of tx and ty (e.g. length of input and response) may differ, and $x_1 \dots x_{tx}$ is the input sequence. At each step, the RNN's hidden state is updated until it reaches x_{tx} and carries information pertaining to the entire string, which we'll call c. The goal of the decoder RNN is the reverse: to predict the response string based on the encoded string created by the encoder, the hidden state created by the encoder, and the final information about the original string c. [46] The hidden function of the decoder is than a function:

$$h_t = f\ (h_{t-1,} y_{t-1,} c) \text{ [46]}$$

Both RNN's are trained to maximize the likelihood of probability that the model outputs the response sequence y, given the input sequence x, as seen in the equation below. [46]

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log\ P_{\theta}(y_n|x_n) \text{ [46]}$$

θ is the model's parameters. The model is trained on real dialogues with <input> <response> pairs. The resulting RNN's could be used for two functions: in SMT, they could be used to generate a <response> translation of an input, which is how they are used here. Second, they could be used to score a response in how well it matches an input, which is how Seq2Seq is used in information retrieval processes. [46]

Cho et. al.'s approach is now the basis for most chatbot SMT response generation, but it is built on a long history of neural network-based SMT techniques. Schwenk et. al. in 2012 used feed-forward neural networks with an input capped at 7 words to generate responses. [48] Devlin et. al. in 2014 advanced the use of feed-forward neural networks, but maintained the necessity of capping the input length a priori. Seq2Seq is widely used in response generation because it does not require a set input length, and achieves stronger performance than earlier models. [49]

While the model is effective, however, Seq2Seq has a number of serious drawbacks. First, Seq2Seq trains the encoder-decoder complex to generate the best response to a single <input> by maximizing the log-likelihood conditional probability: [46]

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log\ P_{\theta}(y_n|x_n) \text{ [46]}$$

While this function offers an approximation of a good response, it does not achieve the chatbot's true goal: to simulate human-human interaction by "providing interesting, diverse, and informative feedback that keeps users engaged." For example, bots will frequently respond to inputs with the phrase "I don't know" because of the frequency with which this response is used in training sets. Second, Seq2Seq often gets stuck in infinite loops, as shown by Li et. al. in simulations of conversations between two Seq2Seq bots. [50]

3.3.5. Reinforcement Learning with Seq2Seq bots

To solve these problems, we must refine the objective of our chatbots by defining long-term reward functions that encourage bots to take steps that will lead to good conversations – those that are "forward-looking…interactive, informative, and coherent." [51]

We can do so using reinforcement learning. Reinforcement Learning (RL) approaches define dialogue as a Markov Decision Process with a state space S, transitions T, action set A, and reward function R. Specifically, because of the possibility of errors in ASR and throughout the process, we consider a Partially Observable MDP or POMDP since we are uncertain of the state space. In POMDPs, we encode states as b(s), our belief about the state, rather than state itself. This allows the system to recover from errors by shifting its beliefs between different states throughout the dialogue. [51]

In this case, the bot's state space b(s) is based on what previous dialogue the bot has heard and used in its past engagement. Our action set A is a set of possible responses (e.g. the RNN algorithm will generate a number of possible responses. Typically, we select the best-fit response, but in this case, we select the top N responses, and call those the actions which the system can select). The chatbot's goal is to define the optimal response strategy, where the number of possible responses is theoretically infinite (we allow for arbitrary length strings). [50]

The reward function is a weighted average of reward functions: $r = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3$, where $r_{1,2,3}$ are defined below and $\lambda_{1,2,3}$ are .25,.25,.5 in Li et. al.'s approach. [50]

Ease of Answering: To facilitate extended dialogue, we would like to reward the bot for statements that are easy to respond to. Li et. al. creates a forward-looking reward function that takes this into account by manually creating a list of dull responses (e.g. "I don't know what you are talking about", "I have no idea", etc.). [50] They define:

$$r_1 = -\frac{1}{N_s} \sum_{s \in S} \frac{1}{N_s} \log P_{seq2seq}(s|a) \text{ [50]}$$

s is the list of dull responses, N_s is the number of dull responses, and $P_{seq2seq}(s|a)$ is the probability of encountering a dull response (the idea is that if the probably of achieving these representative dull responses is low, the chance of generating dull responses overall is low). [50]

Information Flow: The second problem Li et. al. identified was repetition in answers: this reduces the flow of information between bot and user. To solve this problem, they propose penalizing the bot for generating answers that are too semantically similar to previous responses: the reward function is the negative log of the cosine similarity between two consecutive responses $h_{p_i}$and $h_{p_{i+1}}$. [50]

$$r_2 = -\log\cos\left(h_{p_i}, h_{p_{i+1}}\right) = -log\frac{h_{p_i} * h_{p_{i+1}}}{\left\|h_{p_i}\right\| * \left\|h_{p_{i+1}}\right\|}. \text{ [50]}$$

Semantic Coherence: To ensure that the responses are grammatical and coherent, Li et. al. design a third and final reward function, which is the probability of generating this response given the previous dialogue p_i, q_i, which we presume is coherent, plus the reverse probability of generating the question q_i if we can run reverse seq2seq on our proposed response. [50] They define, where a is the proposed response and N_a and N_{q_i} are the cardinalities of a and q_i:

$$r_3 = \frac{1}{N_a} \log p_{seq2seq}(a | q_i, p_i) + \frac{1}{N_{q_i}} logp_{backwardSeq2Seq}(q_i | a) \text{ [50]}$$

To find the optimal policy, reinforcement learning is run in a simulation between two Seq2Seq bots. The resulting bots (now running the optimal policy) are compared against Seq2Seq bots without reinforcement learning, and judged by crowdsourced workers. The results indicate that in a single-turn, reinforcement learning does not improve performance (40% wins for RL-based, 36% for non-RL-based bots). However in multi-turn measurements, for which the RL algorithm was built, RL bots win 72% of the time. [50]

3.4. Knowledge Base Creation

Chatbots are only as intelligent as the knowledge they have access to. Collecting training data used to train machine learning classifiers used in generative bot models or building corpuses of data used by information retrieval bots is critical to achieving human-like interactions.

Advancing the corpora of data used by bots is a complement to algorithm development in improving bot accuracy. This sections focuses on methods for data collection for chatbots. Initially, for bots like ALICE developed in the 20th century, rule-based approaches were complemented with manually created knowledge bases. Later, developers began to use large human-annotated dialogue data-sets. Recent years have seen a rise in the scraping of millions of online dialogues and automated creation of knowledge-bases used by chatbots.

3.4.1. Human-Annotated Corpora

Recognizing the value of already-existing dialogue corpora, Shawar et. al. and others have developed algorithms to convert human-annotated dialogues into AIML, and other chatbots-compatible formats. Their programs have four basic steps: (i) read the dialogues from the existing corpora, (ii) remove all overlapping text (e.g. two speakers talk at once) and non-verbal fillers, (iii) process the text to consider every first line in a dialogue as the <pattern> and the second line as the <template>, and (iv) turn these dialogues into AIML files, or another chatbots-compatible formats so that these dialogues can be used to form the basis of knowledge bases. [52]

Shawar et. al. used the Dialogue Diversity Corpus (DDC) as the basis for their information extraction. The DDC is composed of six corpuses. (i) MICAS is a collection of dialogues recorded and transcribed from academic speech events at the University of Michigan; it includes long monologues which are ineffective for bot-training and non-verbal cues (e.g. “LAUGH”). (ii) CIRCLE was collected by the Center for Interdisciplinary Research on Constructive Learning Environments and contains dialogues from tutorials on physics, algebra, and geometry; it does not contain consistent formatting and is difficult to parse. (iii) CSPA is a set of transcripts from conversations recorded between 1994 and 1998 across a variety of topics; like MICAS, it contains extensive monologues. (iv) TRAINS is a collection of task-oriented spoken dialogues; it is well-formatted, but contains non-verbal annotations. (v) ICE-Singapore is a collection of Singapore English; it also contains non-verbal annotation. (vi) Finally, Mishler Book is a transcription of medical interviews; it is in image-format. [52]

The problems with each of these dialogues necessitated the use of hand-crafted filtering and processing by Shawar et. al. This is generally an issue with human-annotated corpora: most were not designed for the purpose of use by chatbots, and are not formatted in useful ways. The corpora discussed are English language-based because English is the predominant language used in NLP research; however, similar corpora exist in other languages. Conversion processes to AIML have been implemented or attempted in French, Afrikaans, Spanish, and Arabic. [52]

3.4.2. Discussion Forums

Huang et. al. generated the first chatbot knowledge forum scraped from online discussion forums. The benefit of using forums is that they contain different discussion theme pages, giving bot developers access to dialogue on a wide range of topics. Second, these dialogues may involve multiple responses to the same question, which the bot can treat as separate <input><response> pairs. The disadvantage is that discussion forums are a type of asynchronous communications; users do not respond in real-time to each other's queries as required by bots, and this may affect communication style. Second, quality control is a serious issue: replies on forums are edited, short, and may be filled with errors. Third, there is no guarantee that the <response> is a reply to the previous <input>, and not a response to an earlier <input>. [53]

To solve for these deficiencies, Huang et. al. define a high-quality direct reply on a thread (called an RR). They use quality-labeled threads as training data to train a support vector machine (SVM) classifier that can, after training, be used to predict the quality of postings. [53] SVMs are appropriate for the task because (i) this is a binary classification problem identifying quality vs. non-quality RRs and (ii) SVMs are robust to over-fitting. [54]

Huang et. al. use the following features in the SVM model: (i) does reply and quotes root message, (ii) does reply and quotes other replies, (iii) is reply posted by root poster, (iv) # of replies between this reply and previous reply by same author, (v) # of words, (vi) # of content words, and (vii) # of overlapping words between reply and thread-title, among other features. Huang et. al. use author-related features, extracting information from users' forum profiles such as # of threads started by the author, to assess credibility of the reply based on its author. [53]

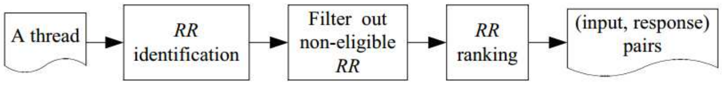


Figure 12: Model for SVM Classification of RR Replies [53]

After classification, three filters were applied to: (i) remove replies that use words on an obscenity list, (ii) remove personal replies that use the phrase "my" (these will not apply to the bot), and (iii) remove forum-specific terms. Based on the SVM classifier and after filtering, 53% of replies were classified as high-quality RR, and these were <thread title> <response> pairs were used to train the chatbot in question. [53]

3.4.3. Email Conversations

Shrestha et. al. use email conversation to extraction question-answer pairs. This approach benefits from the prevalence of email conversations, but has a number of drawbacks including: (i) long monologues used, (ii) emails are asynchronous unlike chat, (iii) stylistic differences in chat vs. email communication and (vi) lack of clarity as to the specific question-answer pairs within longer emails. Only the last of these challenges is prohibitive, and solving this challenge is the focus of Shrestha et. al.'s research. [55]

Shrestha et. al. start by identifying questions in emails by building a machine learning classifier trained on sentences from the SWITCHBOARD corpus and annotated with DAMSL tags which indicate whether sentences are "statement-opinion", "statement-non-opinion", "yes-no-question", and "rhetorical-question" (the corpus and tags are discussed in the Dialogue Act section). [55]

In prediction, a number of features are used including: (i) part of speech (POS) for the first five terms, (ii) POS for the last five terms, (iii) length of utterance, and (iv) POS-bigrams in the sentence from a list of 100 discriminating POS-bi-grams (POS-identification is discussed in the Information Extraction section). The model used for classification is the Ripper program which takes in output classes, features, and training data, and outputs a set if-then questions to be used in prediction. The results led to 72% recall, 96% recall, and F_1score of 0.82. Question extraction from emails achieve better results than from speech because of written indicators (e.g. question mark). [55]

Given an email, we can now find questions. Step two is, given a reply, to identify the answer. In training, human subjects label the Q-A pairs. These are used in a classifier, which keeps track of features, including: number of non-stop words in question and answer segments and (ii) cosine similarity and Euclidian distance between question and answer segments, among other features. In testing, the classifier takes in a question and the <reply> email, and is able to identify the answer based on its knowledge of the relationship between questions and answers, learned during training. Overall, this leads to ~70% precision and 62% recall. [55]

3.5. Dialogue Management

Once the chatbot has selected a response, the Dialogue Manager (DM) must choose a number of communication strategies, use language tricks to make it seem human, and deliver the message.

3.5.1. Communication Strategies

The first step in the DM system design is to choose an interaction strategy, which determines who is leading the conversation: the user, the system, or both. When user-directed initiative is used, the user initiates dialogue, and the system returns responses. The problem with this approach is that the user may provide ill-defined inputs, to which the system does not know how to respond. When system-directed initiative is used, the system directs the dialogue and user responds to queries. This significantly reduces the amount of undefined responses from the user, but reduces the flexibility of the system and restricts the behavior of the user. The mixed-initiative strategy allows the user and system to lead dialogue in different contexts. [16]

The second step is to choose an error handling and confirmation strategy. With explicit confirmation, the system repeats back to the user the query it thinks it heard. McTear, Callejas, and Griol (2016) provide the following example:

User: “I want to go to New York.”
System: “You want directions to New York?

This is time consuming, but ensures the correct response. With implicit confirmation, the system includes what it thinks it heard back to the user as part of the next question.

User: “I want to go to New York.”
System: “What type of transportation do you want to use to get to New York?” [16]

Reinforcement learning, discussed at length in the response generation section, can be used in a similar way to train bots to find optimal interaction and confirmation strategies. Singh et. al., for example, use reinforcement learning to find the optimal policy for communication style. They define the possible actions in their action set as: (i) user directive – open question with non-restrictive grammar in the ASR, (ii) system directive – directive question with restrictive grammar, or (iii) mixed initiative – directive question with unrestrictive grammar. Second, Singh et. al. use reinforcement learning to find the optimal policy for confirmation: explicit confirmation or no confirmation. The reward functions for these learning tasks were part of the experimental design: subjects were given scripts, and these scripts corresponded to desired behavior from the system. If this behavior occurred, the system received a reward of 1. Otherwise, it received a reward of 0. Singh et. al. found that reinforcement learning enabled the system to improve over time, and converge on the optimal policies for communication style and confirmation. [57]

Likewise, active learning can be used to improve the bot’s ability to generate effective responses. As discussed, response selection typically generates a list of ranked responses, the highest ranked of which is selected. In active learning, the user is shown a number of responses (e.g. the top five) at each turn, and selects the preferred response. This allows the system to learn in real-time, and adapt to the specific needs of each user. There are two costs, however. First, this is time-consuming and distracting for the user, especially if the bot is being used for an important application (e.g. mHealth application). Second, the average bot user is not an expert in DM design strategy, and therefore may not actually be able to choose the appropriate response. [58]

3.5.2. Language Tricks

Many of the response generation strategies previously discussed are generated alongside confidence scores that indicate how likely the bot is that its response is appropriate. For responses that have low probabilities of being appropriate, bot developers employ a number of “language tricks” that improve conversation outcomes:

Yu et. al. [56] discusses a number of these strategies used by the bots they develop, including:

A. **Switch Topic:** Chatbot proposes a new topic, for example talking about sports, the weather, or a number of other pre-programmed, buffer topics. While this can generate non-sequiturs, users are more likely to rate these responses as appropriate than the low-confidence responses otherwise generated.

B. **Ask Open Questions:** Rather than simply saying “I don’t know” when the bot cannot understand the input, it says “Sorry, I don’t know. Could you tell me something interesting?" This is effectively the same answer, but gives the user a next step to continue the conversation.

C. **Tell a Joke:** This may be a non-sequitur, but continues the conversation.

D. **Elicit more information:** If the chatbot doesn't have an adequate response, it may simply ask the user to provide more information. This is an open-ended question that continues the conversation and gives the bot another chance at interpreting the new input it will receive.

3.5.3. Dialogue Design Principles

Chatbots provide these functions through a dialogue interface, and are developed to adhere to a number of dialogue design principles including: (i) disambiguation – bots clarify user inputs when they could have multiple meanings (e.g. input: "Was Amadeus nominated for an academy award?" and response: "Is Amadeus the movie's title?"), (ii) relaxation – ability to drop constraints to properly handle user inputs (e.g. input: "Are there any flights today" and response: "No. Would you like me to find one tomorrow?" (iii) confirmation – checking user details when a task could not be carried out or for import decisions (e.g. response: "you want me to book you the 7am flight from Philadelphia to San Francisco?"), and (iv) completion – asking for necessary details left out in a user request (e.g. input: "I would like to book the 7am flight to SF" and response: "Which airport are you flying out of?"). [59]

3.5.4. Human Imitation Strategies

There a number of different strategies that can be employed to make bots appear more human. Pereira et. al. [60] suggests a few common approaches:

A. **Personality Development**: Most chatbots from ELIZA to ALICE and CHARLIE have names, as well as related personalities in the hopes of making them appear more human. ELIZA's DOCTOR script took on the personality of a Rogerian psychologist, while PARRY took on the personality of a paranoid mental patient (which enabled it to get away with non-sequitur answers).

B. **Direct the Conversation:** ELIZA commonly asked the user questions to elicit user participation and minimize the chance for error. This type of approach is frequently used in chatbots to make them appear more human and was used by Converse in 1997 to successfully convince a Loebner prize judge that it was a human, for the first five minutes of conversation.

C. **Small Talk:** Faced with task-specific situations, humans revert to "neural, non-task oriented conversation about safe topic, where no specific goals need to be achieved." For bots to appear human, developers build in small-talk functionality which builds rapport with the human, and avoids silence (if the human is not initiating dialogue).

D. Human-like Failures: Chatbots often contain flaws, and while too great a propensity to make mistakes does not bode well, flawless communication may also indicate that the bot is not human. Including disfluencies in dialogue makes bots appear human.

Hung et. al. introduce the notion of goal-orientation as a mechanism for making bots appear more human. They argue that to truly mimic humans machines must be able to understand goals, and propose a framework in which chatbots engage in goal recognition, goal bookkeeping, and context topology. Using many of the NLP methods previously discussed including keyword extraction and POS tagging, bots use linguistic analysis to infer the goal of the human with which they are speaking at each turn of speech, and add these goals to a goal stack. Goals are associated with specific contexts, and bots use context-based reasoning (CxBR) to direct their responses; as goals change, contexts change, and bots communication styles may change. [61]

3.5.5. Personality Development

Personality infusion into chatbots, at its most basic level, can be accomplished directly by manually creating the <pattern><template> pairs, and incorporating personality features across the <templates>. This was the strategy employed by ALICE (e.g. its DOCTOR script). This makes the personality hard to change, however, unless an entirely different script is used.

Galvão et. al. propose an alternate model in which a "personality component" exists outside the knowledge base. Particular personality elements can be selected for each different bot (or instance of the same bot), or based on the dialogue using natural language processing. Galvão et. al. propose tying rules to personality elements, so that as personality changes so do the bot's selections of responses. Their bot, for example, "only answers personal questions in case it is happy" and "only searches the Internet for answering users' doubts in case he likes the user." [62]

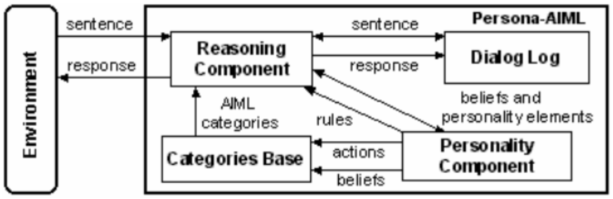
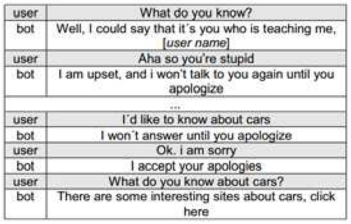


Figure 14: Persona-AIML Model [62]



user	What do you know?
bot	Well, I could say that it´s you who is teaching me, [*user name*]
user	Aha so you're stupid
bot	I am upset, and i won't talk to you again until you apologize
...	
user	I´d like to know about cars
bot	I won´t answer until you apologize
user	Ok. i am sorry
bot	I accept your apologies
user	What do you know about cars?
bot	There are some interesting sites about cars, click here

Figure 13: Example bot response based on emotions [62]

In practice, emotions are tied to responses in two ways. First, the system adopts a different set of rules depending on its emotions or mood. This can be modeled as separate knowledge bases accessed for each emotion, or as <category> <template1> <template2>… <templateN> sets, where each template corresponds to one of N emotions. Second, each <template> response contains one or more emotion associated with it, each with a probability, such that depending on where the conversation goes, the bot will change its emotions. When emotional tags are selected per the probability distribution, they are passed to a personality module, which can be implemented as a Bayesian Belief Network (BBN). The personality module will, given a current emotion and a new emotional-tag change the bots "mood." [63]

In Thalmann's implementation, changing personality/emotion changes the facial expression of the avatars of bots. For bots without physical representations, use of punctuation (e.g. exclamation) and emoticons may be used to express emotion as well. [63] Personality/emotion could also change the speed or volume of speech in a speech-based chatbot system. [64]

Responses do not have to change radically based on emotions, as in the previous implementations. Ball et. al. propose using paraphrases for existing speech based on emotional state, such as saying "if you insist," rather than "yes" when frustrated. Likewise, they propose using different paraphrases based on the specific bot (or bot instance's) personality, such as choosing between "you should definitely" or "perhaps you might like to." While the discussion of emotion-based response has so far been focused on entertainment, chatbots can also incorporate task orientation to improve the user experience. For example, if speech is rushed, the bot could shift its personality to terse, and give short replies. [64]

3.6. Text to Speech

Text-to-Speech (TTS) is the final step of the process, and converts the response generated to speech, which is returned back to the user. The first step of TTS is text analysis, in which text is converted into phonemes with pitch and duration. The second step is waveform synthesis, in which segments of recorded speech corresponding to each phoneme are concatenated to form speech.

3.6.1. Text Analysis

Text Analysis begins with a normalization process in which text is broken up into its component parts; chunks of text are broken into sentences, sentences into words and punctuations, and words into their respective phonemes. Sentence tokenization involves searching for typical sentence break punctuation, such as the exclamation mark, question mark, colon, semi-colon, or period. The challenge is that these punctuations do not always mark the end of a sentence; to solve this problem, machine learning classifiers are trained on text to learn the difference between sentence-ending and non-sentence-ending punctuations. [65]

Normalization also includes the conversion of non-standard words into natural language. Examples include numbers, abbreviations, and acronyms which can be pronounced in a number of different ways. Finally, normalization involves the disambiguation of homographs, or words that are pronounced differently and contain different meaning based on context. For these words (e.g. "I used by car" vs. "I put my car to use"), part-of-speech (POS) analysis can be run to tag the part of the speech of the homograph, and in doing so, disambiguate its pronunciation. [65]

The second step in text analysis is pronunciation. This involves access to a pronunciation lexicon which includes mappings between words/phonemes and their pronunciations, as well as name lexicons, which map names to their pronunciations. The character-to-phoneme mapping process was initially rule-based, but has since progressed and now uses advanced algorithms to generate the most accurate mappings based on context (there is no one-to-one mapping). [65]

Pagel et. al. used decision trees for the process [66] and Sejnowski and Rosenberg trained a feed-forward neural network. [67] Pronunciation alone would lead speech to sound like a string of words. Prosody information is needed to add proper “phrasing, prominence, and intonation,” and a number of machine learning classifiers have been developed and trained to provide proper mapping between characters and prosidy information. [65]

3.6.2. Waveform Synthesis

Wavelength synthesis is the voicing of the sounds generated in text analysis, and can occur using a number of different processes. Formant synthesis involves a number of resonators connected either in series or parallel that pass the output of frequency production one to the next (typically three is required for understandable sound, and five for high-quality sound production). Articulatory synthesis theoretically is intended to generate “speech by direct modeling of the human articulator behavior,” but does not produce high-quality results in practice.

Concatenative synthesis, by contrast, generates speech by connecting a number of pre-recorded speech units. The most common unit used is the di-phone which starts at the middle of one phoneme and extends to the middle of a second, capturing both sounds and the transition. The problem with concatenative synthesis is the prosidy of units may not align. To solve this problem, unit selection synthesis stores multiple instances of each unit, each with a different prosidy, and selects the unit to deliver based on the context. Storing multiple instances of each unit can be space-inefficient. Statistical models (e.g. HMM-synthesis models) exist to solve this problem by mapping from one instance onto the variant with the correct prosidy. [65]

4. Applications & Development

4.1. IBM Watson Case Study

IBM's Watson gained popular recognition for its Jeopardy! win. The architecture behind Watson is a Question-Answer chatbot; in this section we take a deep dive into its workings.

4.1.1. Natural Language Processing

The first step is Question Analysis, in which Watson runs natural language processing techniques to classify the problem, including shallow parsing, deep parsing, semantic role labeling, conference relations, and named-entity recognition, most of which are discussed in the section on natural language processing (NLP). [68] Deep parsing techniques in Watson involve taking any input sentence and performing tokenization and segmentation, followed by a number of linguistic analyses (e.g. morpholexical and syntactic analysis). Tokens are converted into one-word phrases, and then further processed into sequences of multiple word phrases that carry meaning. Through this process, Watson is able to break down a sentence into its component parts, label each, and understand the functional and grammatical significance of each for use in response generation. [69]

4.1.2. Response Generation

The second step is Hypothesis Generation. Based on its understanding of the question, Watson searches its external knowledge bases for an answer, and generates all possible content that could contain the answer. This "primary search" phase is focused on maximizing recall, and assumes later analytical work will be completed to glean the correct response. Watson's goal is to achieve 85% recall for the top 250 candidates (e.g. for 85% of questions, the correct answer will be contained in one of the top 250 candidate responses retrieved). [68]

Search techniques include: text search engines, knowledge base search, document search, and passage search and include multiple search queries per question. In document search, Watson extracts the titles of documents that contain the text of the search query as hypotheses. In passage search, named-entity recognition techniques are used on passages with the text of the search query, and these named-entities are generated as hypotheses. Passage search and document search use existing platforms such as Lucene and Indri to find relevant documents that may contain the answer to the search query. [68]

The third step is hypothesis and evidence scoring, in which the system collects evidence on each hypothesis, and applies a ranking algorithm to determine the most likely response. Scoring functions measure: "the degree of match between a passage's predicate-argument structure and the question, passage source reliability, geospatial location, temporal relationships, taxonomic classification, the lexical and semantic relations the candidate is known to participate in, the candidate's correlation with question terms, its popularity (or obscurity), its aliases, and so on." For example, Watson uses a passage score that calculates IDF-weighted terms in common between the input and hypothesis, one that measures the longest common subsequence between input and hypothesis, and a third that measures the alignment of logical forms in input and hypothesis. [68]

Watson likewise is developed to include reasoning capabilities. For example, Watson is developed to contain temporal reasoning, such that it can detect inconsistencies in dates and geospatial reasoning such that it can detect inconsistencies in geographies (e.g. Sydney is not an Asian city). All of the aforementioned scorers are aggregated into dimensions on which each response is scored, such as "Taxonomic, Geospatial (location), Temporal, Source Reliability, Gender, Name Consistency, Relational, Passage Support, Theory Consistency, and so on." [68]

The fourth step is ranking and confidence estimation. Watson uses a machine learning approach to achieve this step; it takes in the above dimensions as features and trains a classifier on training data with known correct answers, thus creating weights for each dimension. Given the variety of dimensions, different classifiers are applicable to each. To solve this problem, Watson trains separate classifiers to handle sub-sets of features, and then uses an ensemble of these classifiers to generate an overall ranking. [68]

4.1.3. Knowledge Base

Watson's knowledge, primarily, is generated through access to a collection of online documents through two platforms: Indri and Lucene. Accessing these platforms, Watson is able to pair queries with document sets, and extract information containing the answer either from the titles of the documents of from the contents of the documents. Typical bots run this same procedure using the Google search engine, Bing search engine and Wikipedia; Watson did not in this case because it was run offline as part of the Jeopardy! rules. [68]

Indri is an open source search engine developed out of a collaboration between the University of Massachusetts and Carnegie Mellon University. Indri was built on top of the Lemur project, a toolkit built for language modeling and information retrieval. [70] Indri is widely used because it uniquely allows for "complex phrase matching, synonyms, weighted expressions, Boolean filtering, numeric (and dated) fields, and the extensive use of document structure (fields)." [71]

Lucene is likewise a text search engine library. The program was built by the Apache Software Foundation, and is used as an open source data source. [70] Lucene provides ranked searching, "phrase queries, wildcard queries, proximity queries, range queries, fielded searching, multiple-index searching with merged results, [and a]… configurable storage engine." [72]

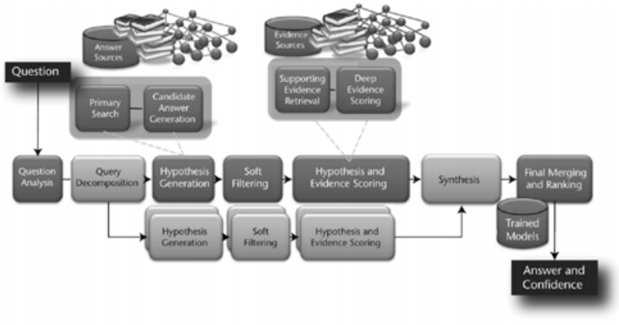


Figure 15: Watson Retrieval Process [68]

4.2. Security Considerations

Chatbots are widely used in a variety of application including mobile health (mHealth). Both for regulatory and ethical reasons, then, privacy of chatbots (or lack thereof) is a significant design consideration, and potential architecture vulnerability. [74]

4.2.1. Security Flaws in Chatbot Platforms

Chatbots are primarily accessed through chat/messenger applications (e.g. Facebook, Skype, etc.). This is problematic from a security perspective given the vulnerabilities of these systems.

Based on the Electronic Frontier Foundation (EFF) Secure Messaging Scorecard, Facebook Messenger, for example, is not secure in five of seven tested dimension. First, communication is not encrypted by default, which means that Facebook, or a government agency, can read these messages. More importantly, failure to use end-to-end encryption could allow individuals that share a local wireless network to sniff each other's credentials, access their accounts, and read their private conversations. Second, Messenger does not guarantee identity confirmation. Third, past communications are not hidden, and can be stolen if a malicious attacker steals user credentials. Fourth, the code is not open to independent review. Fifth, the security design is not properly documented. These vulnerabilities extend to bots utilizing the Messenger platform. [75]

Chatbots themselves have serious privacy implications as well. Users can order an Uber, buy plane tickets, and make payments through Messenger bots. Personal data is stored on the Messenger platform, and often sent, unencrypted, to external tools such as wit.ai, api.ai, and IBM Watson for NLP analysis. More importantly, bot creators themselves have access to personal information inputted into applications, including pictures and forms of identification. [75] Speech-based bots such as Amazon Echo collect even more information: whenever the "wake word" (e.g. Alexa) is used, Echo records the conversation and stores it on the cloud. This again is problematic as consumers are exposing themselves to the possibility of a privacy breach. [76]

4.2.2. Malicious Chatbots

While the above privacy violations are un-intended by the bot creators, chatbot users expose themselves to the possibility of intentionally malicious chatbots, created to exploit their users. Already, there exist an extensive number of spam-bots that exist to send spam to internet users. While these bots are detectable, and often shut down, the improving conversation abilities of bots will make it easier for malicious bots to disguise themselves. These chatbots, able to build trust with users, may be programmed to exploit that trust, sell products, infect computer with malware, and steal identity information. [77] Lauinger et. al.'s chatbot HoneyBot, for example, achieved a click-through rate of 37% sending phishing messaging using a man-in-the-middle attack. [78]

4.3. Applications

4.3.1. Virtual Personal Assistants (VPAs)

Virtual Personal Assistants (VPAs) are among the popular applications of the chatbot technology. In particular, Apple Siri, Microsoft Cortana, Google Assistant, and Amazon Alexa have emerged to offer users services over speech and text interfaces.

Apple Siri: Siri is a virtual assistant specifically available on Apple products, and has access to Apple-applications including Mail, Contacts, Messages, Maps, and Safari. Siri can read users' email, text contacts, change music playing, make calls, find restaurants, find books, set alarms, and give directions. Siri's gender, accent, and language are configurable and changeable. [79]

Microsoft Cortana: Cortana can be used for basic search functionality (e.g. search for the weather), can access users' google calendars, read their Outlook emails, give time estimations for travel, give directions, and integrate with OneNote to show users' their notes. [80]

Google Assistant: Google Assistant is an extension of the basic "OK Google" functionality that allows users to conduct search and control their mobile devices through voice commands. Assistant is programed into Google phones, Android OS and will be integrated into some cars. [81]

Amazon Alexa: Alexa can access the weather, connect to radio and television stations, and has partnerships with a number of services, including: JustEat, Uber, FitBit, The Telegraph, Spotify and Nest, among others. These services can be accessed with the Alexa interface. [82]

4.3.2. Consumer Domain-Specific Bots

Transportation Bots: Instalocate provides real-time flight tracking, security wait times, ability to file for compensation for delayed flights, ability to push notifications to WhatsApp, gate and terminal information, weather updates, airport directions, baggage information, and an ability to book cabs or Uber vehicles at the airport. The bot uses rule-based heuristics for responses; conversation is restricted to system-directed initiatives. [83]

Dating: Foxsy suggests new friends to you based on user-specified settings. Effectively a Tinder-like applications embedded within the Messenger platform. The bot uses a very basic rule-based heuristic for responses; allows for user-directed conversation, but replies with "Sorry, I didn't get it" to most queries. [84]

Mediation: Meditate Bot provides users with information about breathing exercises and body scans and allows users to schedule daily medications. The bot notifies users when it is time for their daily medications. [85] Peaceful Habit offers similar functionality in scheduling and timing meditation routines for its users. [86]

Fitness Bots: GymBot allows users to keep track of the number of exercises they've completed and track their improvement over time. [87] FitCircle generates five-minute workouts for users. Forksy asks you about what food you're eating, and scolds you for unhealthy choices. Whole Foods bot allows you to search for healthy recipes with text or emoji requests. [88]

Weather Bots: Poncho is a weather bot with a distinctive personality that allows users to request the current weather conditions. [89] The Weather Channel offers a more traditional weather bot, powered by IBM Watson. [90] Other weather bots include: HippoBot, Weatherman, and Yahoo weather; they provide similar functionality with differing personality settings. [91]

Medical Bots: The symptoms defining medical conditions are often well-defined, and chatbots have the ability to improve rates of accurate self-diagnosis. One of the main problems with self-diagnosis is that most individuals do not know what questions to ask in order to diagnose a medical condition, and therefore are unable to use platforms like WebMB properly. Chatbots, which are able to ask follow-up questions until they are ready to make a diagnosis, solve that problem. Baidu, the Chinese search engine, for example, launched a diagnosis app called Melody in 2015, which responds to medical questions from users. [92]

5. References

[1] Jia, J. (2003). The Study of the Application of a Keywords-based Chatbot System on the Teaching of Foreign Languages. ArXiv preprint cs/0310018. Retrieved from https://arxiv.org/abs/cs/0310018

[2] Sansonnet, J.-P., Leray, D., & Martin, J.-C. (2006). Architecture of a Framework for Generic Assisting Conversational Agents. Intelligent Virtual Agents Lecture Notes in Computer Science, 145–156. http://doi.org/10.1007/11821830_12

[3] Turing, A. M. (1950). Computing Machinery And Intelligence. Mind, LIX(236), 433–460. http://doi.org/10.1093/mind/lix.236.433

[4] Weizenbaum, J. (1966). ELIZA---a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36–45. http://doi.org/10.1145/365153.365168

[5] Colby, K. M. (1981). Modeling a paranoid mind. Behavioral and Brain Sciences, 4(04), 515. http://doi.org/10.1017/s0140525x00000030

[6] Wallace, R. S. The Anatomy of A.L.I.C.E. Retrieved from http://www.alicebot.org/anatomy.html

[7] Weinberger, M. (2017, January 14). Why Amazon's Echo is totally dominating - and what Google, Microsoft, and Apple have to do to catch up. Retrieved from http://www.businessinsider.com/amazon-echo-google-home-microsoft-cortana-apple-siri-2017-1

[8] Jurafsky, D. Conversational Agents. Retrieved from http://web.stanford.edu/class/cs124/lec/chatbot.pdf

[9] A framework for chatbot evaluation. (2017, January 24). Retrieved from https://isquared.wordpress.com/2017/01/24/a-framework-for-chatbot-evaluation/

[10] Walker, M. A., Litman, D. J., Kamm, C. A., & Abella, A. (1997). Paradise. Proceedings of the 35th annual meeting on Association for Computational Linguistics -. http://doi.org/10.3115/976909.979652

[11] Bates, M., & Ayuso, D. (1991). A proposal for incremental dialogue evaluation. Proceedings of the workshop on Speech and Natural Language - HLT '91. http://doi.org/10.3115/112405.112472

[12] Hung, V., Elvir, M., Gonzalez, A., & Demara, R. (2009). Towards a method for evaluating naturalness in conversational dialog systems. 2009 IEEE International Conference on Systems, Man and Cybernetics. http://doi.org/10.1109/icsmc.2009.5345904

[13] Abdul-Kader, S., & Woods, J. (2015). Survey on Chatbot Design Techniques in Speech Conversation Systems. International Journal of Advanced Computer Science and Applications, 6(7). http://doi.org/10.14569/ijacsa.2015.060712

[14] Nass, C., & Brave, S. (2007). Wired for speech: how voice activates and advances the human-computer relationship. Cambridge, MA: MIT Press.

[15] Gruhn, R. E., Minker, W., & Nakamura, S. (2013). Statistical Pronunciation Modeling for Non-Native Speech Processing. Berlin: Springer Berlin.

[16] McTear, M., Callejas, Z., & Griol, D. (2016). The conversational interface: talking to smart devices. Cham: Springer.

[17] Mohamed, A.-R., & Hinton, G. (2010). Phone recognition using Restricted Boltzmann Machines. 2010 IEEE International Conference on Acoustics, Speech and Signal Processing. http://doi.org/10.1109/icassp.2010.5495651

[18] Tur, G., & Mori, R. D. (2011). Spoken language understanding: systems for extracting semantic information from speech. Hoboken, NJ: Wiley.

[19] Kral, P., & Cerisara, C. (2010). Dialogue Act Recognition Approaches. Computing and Informatics, 29, 227–250. http://doi.org/10.1109/mlsp.2008.4685529

[20] Draft of DAMSL: Dialog Act Markup in Several Layers*. Retrieved from http://www.cs.rochester.edu/research/cisd/resources/damsl/RevisedManual/

[21] WS-97 Switchboard DAMSL Coders Manual. Retrieved from https://web.stanford.edu/~jurafsky/ws97/manual.august1.html

[22] Dhillon, R., Bhagat, S., Carvey, H., & Shriberg, E. (2004, February 9). Meeting Recorder Project: Dialog Act Labeling Guide. Retrieved from http://www1.icsi.berkeley.edu/ftp/pub/speech/papers/MRDA-manual.pdf

[23] Carletta, J., Isard, S., Doherty-Sneddon, G., Isard, A., Kowtko, J. C., & Anderson, A. H. (1997). The Reliability of a Dialogue Structure Coding Scheme . Association for Computational Linguistics, 23(1), 13–31. Retrieved from http://www.aclweb.org/anthology/J97-1002

[24] Reithinger, N., & Klesen, M. (1997). Dialogue Act Classification Using Language Models. In Proceedings of EuroSpeech, 2235–2238.

[25] Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., … Meteer, M. (2000). Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. Computational Linguistics, 26(3), 339–373. http://doi.org/10.1162/089120100561737

[26] Stolcke, A., & Shriberg, E. (1998). Dialog Act Modeling for Conversational Speech. AAAI Technical Report SS-98-01, 98–105. Retrieved from https://pdfs.semanticscholar.org/d7de/4acec556524a8ab2b00aa4414768bc1617e9.pdf

[27] Ries, K. (1999). HMM and neural network based speech act detection. 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258). http://doi.org/10.1109/icassp.1999.758171

[28] Wright, H. (1998). Automatic Utterance Type Detection Using Suprasegmental Features. International Conference on Spoken Language Processing. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.47.5099&rep=rep1&type=pdf

[29] Nielson, M. A. (2015). Chapter 3: Improving the way neural networks learn. In Neural Networks and Deep Learning. essay, Determination Press.

[30] Andernach, T., Poel, M., & Salomons, E. (1997). Finding Classes of Dialogue Utterances with Kohonen Networks. Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing Tasks, 85–94. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.5080&rep=rep1&type=pdf

[31] Rotaru, M. (2002). Dialog Act Tagging using Memory-Based Learning. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.7922&rep=rep1&type=pdf

[32] Jeong, M., & Lee, G. (2006). Jointly Predicting Dialog Act And Named Entity For Spoken Language Understanding. 2006 IEEE Spoken Language Technology Workshop. http://doi.org/10.1109/slt.2006.326818

[33] He, Y., & Young, S. (2005). Semantic processing using the Hidden Vector State model. Computer Speech & Language, 19(1), 85–106. http://doi.org/10.1016/j.csl.2004.03.001

[34] Introduction to Support Vector Machines¶. Retrieved from http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[35] Lokman, A. (2010). One-Match and All-Match Categories for Keywords Matching in Chatbot. American Journal of Applied Sciences, 7(10), 1406–1411. http://doi.org/10.3844/ajassp.2010.1406.1411

[36] AbuShawar, B., & Atwell, E. ALICE Chatbot: Trials and Outputs. Retrieved from http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462015000400625

[37] Yan, Z., Duan, N., Bao, J., Chen, P., Zhou, M., Li, Z., & Zhou, J. (2016). DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). http://doi.org/10.18653/v1/p16-1049

[38] Ganguly, D., Roy, D., Mitra, M., & Jones, G. J. (2015). Word Embedding based Generalized Language Model for Information Retrieval. Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '15. http://doi.org/10.1145/2766462.2767780

[39] Wu, Y., Wu, W., L, Z., & Zhou, M. (2016). Sequential Match Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots. ArXiv preprint 612.01627. Retrieved from https://arxiv.org/pdf/1612.01627.pdf

[40] Chung, J., Gulcehre, C., Cho, K. H., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv:1412.3555v1. Retrieved from https://arxiv.org/pdf/1412.3555.pdf

[41] Statistical Machine Translation. Retrieved from http://www.statmt.org/

[42] Ritter, A. (2011). Data-Driven Response Generation in Social Media. EMNLP '11 Proceedings of the Conference on Empirical Methods in Natural Language Processing, 583–593. Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/mt_chat.pdf

[43] Moses. Retrieved from http://www.statmt.org/moses/?n=Moses.Tutorial

[44] Koehn, P. Phrase-Based Models. Statistical Machine Translation, 127–154. http://doi.org/10.1017/cbo9780511815829.006

[45] Jaccard Similarity and Shingling. Retrieved from https://www.cs.utah.edu/~jeffp/teaching/cs5955/L4-Jaccard Shingle.pdf

[46] Cho, K., Merrienboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). http://doi.org/10.3115/v1/d14-1179

[47] Ramamoorthy, S. Chatbots with Seq2Seq. Retrieved from http://suriyadeepan.github.io/2016-06-28-easy-seq2seq/

[48] Schwenk, H. (2012). Continuous Space Translation Models for Phrase-Based Statistical Machine Translation. Proceedings of COLING 2012: Posters, 1071–1080. Retrieved from https://aclweb.org/anthology/C/C12/C12-2104.pdf

[49] Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., & Makhoul, J. (2014). Fast and Robust Neural Network Joint Models for Statistical Machine Translation. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). http://doi.org/10.3115/v1/p14-1129

[50] Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., & Jurafsky, D. (2016). Deep Reinforcement Learning for Dialogue Generation. ArXiv:1606.01541. Retrieved from https://aclweb.org/anthology/D16-1127

[51] Reiser, V., & Lemon, O. (2011). Reinforcement Learning for Adaptive Systems. Edinburgh: Springer.

[52] Shawar, B. A., & Atwell, E. (2003). Machine learning from dialogue corpora to generate chatbots. School of Computing, University of Leeds. Retrieved from https://pdfs.semanticscholar.org/8bf9/391940801b762de98991c0fb5a426777d104.pdf

[53] Huang, J., Zhou, M., & Yang, D. (2007). Extracting Chatbot Knowledge from Online Discussion Forums. IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence, 423–428. Retrieved from http://www.ijcai.org/Proceedings/07/Papers/066.pdf

[54] Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1–47. http://doi.org/10.1145/505282.505283

[55] Shrestha, L., & Mckeown, K. (2004). Detection of question-answer pairs in email conversations. Proceedings of the 20th international conference on Computational Linguistics - COLING '04. http://doi.org/10.3115/1220355.1220483

[56] Yu, Z., Xu, Z., Black, A. W., & Rudnicky, A. (2016). Strategy and Policy Learning for Non-Task-Oriented Conversational Systems. Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue. http://doi.org/10.18653/v1/w16-3649

[57] Singh, S., Keams, M., Litman, D., & Walker, M. (2000). Reinforcement Learning for Spoken Dialogue Systems. Advances in Neural Information Processing Systems, 956–962. Retrieved from https://papers.nips.cc/paper/1775-reinforcement-learning-for-spoken-dialogue-systems.pdf

[58] Hiraoka, T., Neubig, G., Yoshino, K., Toda, T., & Nakamura, S. (2016). Active Learning for Example-Based Dialog Systems. Lecture Notes in Electrical Engineering Dialogues with Social Robots, 67–78. http://doi.org/10.1007/978-981-10-2585-3_5

[59] Abella, A., Brown, M. K., & Buntschuh, B. (1997). Development principles for dialog-based interfaces. Dialogue Processing in Spoken Language Systems Lecture Notes in Computer Science, 141–155. http://doi.org/10.1007/3-540-63175-5_43

[60] Pereira, M., & Coheur, L. (2013). Just.Chat - a platform for processing information to be used in chatbots. Retrieved from https://fenix.tecnico.ulisboa.pt/downloadFile/395145485809/ExtendedAbstract.pdf

[61] Hung, V., Gonzalez, A., & Demara, R. (2009). Towards a Context-Based Dialog Management Layer for Expert Systems. 2009 International Conference on Information, Process, and Knowledge Management. http://doi.org/10.1109/eknow.2009.10

[62] Galvão, A. M., Barros, F. A., Neves, A. M. M., & Ramalho, G. L. (2004). Adding Personality to Chatterbots Using the Persona-AIML Architecture. Advances in Artificial Intelligence – IBERAMIA 2004 Lecture Notes in Computer Science, 963–973. http://doi.org/10.1007/978-3-540-30498-2_96

[63] Kshirsagar, S., & Magnenat-Thalmann, N. (2002). Virtual humans personified. Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1 - AAMAS '02. http://doi.org/10.1145/544818.544826

[64] Ball, G., & Breese, J. (2000). Emotion and personality in a conversational agent. In Embodied conversational agents (pp. 189–219). essay, Cambridge: MIT Press.

[65] Rashad, M. Z., El-Bakry, H. M., & Isma'il, I. R. (2010). An Overview of Text-To-Speech Synthesis Techniques. CIT'10 Proceedings of the 4th international conference on Communications and information technology, 84–89. Retrieved from https://pdfs.semanticscholar.org/5ee7/c71a362cc01441c27ced052a70ee6e0732dc.pdf

[66] Che, H., Tao, J., & Pan, S. (2012). Letter-to-sound conversion using coupled Hidden Markov Models for lexicon compression. 2012 International Conference on Speech Database and Assessments. http://doi.org/10.1109/icsda.2012.6422464

[67] Sejnowski, T. J., & Rosenberg, C. R. (1988). NETtalk: a parallel network that learns to read aloud. In Neurocomputing: foundations of research (pp. 661–672). essay, Cambridge, MA: MIT Press.

[68] Ferrucci, D. (2010). Building Watson: An Overview of the DeepQA Project. Association for the Advancement of Artificial Intelligence, 31(3), 59–79. Retrieved from http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303/2165\

[69] Mccord, M. C., Murdock, J. W., & Boguraev, B. K. (2012). Deep parsing in Watson. IBM Journal of Research and Development, 56(3.4). http://doi.org/10.1147/jrd.2012.2185409

[70] Middleton, C., & Baeza-yates, R. A Comparison of Open Source Search Engines. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.6955&rep=rep1&type=pdf

[71] Indri Query Language Quick Reference. Retrieved from https://www.lemurproject.org/lemur/IndriQueryLanguage.php

[72] Apache Lucene Core. Retrieved from https://lucene.apache.org/core/

[73] Secure Messaging Scorecard. (2016, September 28). Retrieved from https://www.eff.org/node/82654

[74] Yuan, M. (2016, April 29). Chatbots made for healthcare. Retrieved from https://tincture.io/chatbots-made-for-healthcare-fec631bc8462#.5bmzw2edo

[75] Why you shouldn’t talk to your chatbot about everything. Retrieved from http://venturebeat.com/2016/11/17/why-you-shouldnt-talk-to-your-chatbot-about-everything/

[76] Your Security Resource. Retrieved from http://www.yoursecurityresource.com/expertqa/how-private-is-new-amazon-echo/

[77] Narang, S. (2014, August 4). Are You Talking to a Chatbot? How Spam Chatbots Will Get Smarter. Retrieved from https://www.linkedin.com/pulse/20140804184616-7077712-are-you-talking-to-a-chatbot-how-spam-chatbots-will-get-smarter

[78] Lauinger, T., Pankakoski, V., Balzarotti, D., & Kirda, E. (2010). Honeybot, Your Man in the Middle for Automated Social Engineering. LEET'10 Proceedings of the 3rd USENIX conference on Large-Scale exploits and emergent threats: botnets, spyware, worms, and more, 11–11. Retrieved from https://www.sba-research.org/wp-content/uploads/publications/autosoc-leet2010.pdf

[79] O'Boyle, B. (2015, October 12). What is Siri? Apple's personal voice assistant explained. Retrieved from http://www.pocket-lint.com/news/112346-what-is-siri-apple-s-personal-voice-assistant-explained

[80] Ravenscraft, E. (2015, August 3). Everything You Can Ask Cortana to Do in Windows 10. Retrieved from http://lifehacker.com/everything-you-can-ask-cortana-to-do-in-windows-10-1721725525

[81] Betters, E. (2017, March 2). What is Google Assistant, how does it work, and which devices offer it? Retrieved from http://www.pocket-lint.com/news/137722-what-is-google-assistant-how-does-it-work-and-which-devices-offer-it

[82] O'Boyle, B. (2016, December 26). Amazon Echo: What can Alexa do and what services are ... Retrieved from http://www.pocket-lint.com/news/138846-amazon-echo-what-can-alexa-do-and-what-services-are-compatible

[83] Instalocate. Retrieved from https://www.messenger.com/t/instalocate/

[84] Foxsybot. Retrieved from https://www.messenger.com/t/foxsybot

[85] MeditateBot. Retrieved from https://botlist.co/bots/2021-meditatebot

[86] Peaceful Habit. Retrieved from https://botlist.co/bots/1381-peaceful-habit

[87] GymBot. Retrieved from https://gymbot.io/

[88] Top 5 Bots To Get You Fit. (2016, October 22). Retrieved from http://www.topbots.com/top-5-best-fitness-bots-fitness-apps/

[89] Poncho. Retrieved from https://poncho.is/

[90] The Weather Channel Launches Bot For Facebook Messenger, Powered By IBM Watson. Retrieved from http://www.theweathercompany.com/newsroom/2016/10/25/weather-channel-launches-bot-facebook-messenger-powered-ibm-watson

[91] 12 Weather bots for Facebook Messenger, Slack, Skype and Telegram. Retrieved from https://chatbottle.co/bots/tagged/weather

[92] Vincent, J. (2016, October 11). Baidu launches medical chatbot to help Chinese doctors diagnose patients. Retrieved from http://www.theverge.com/2016/10/11/13240434/baidu-medical-chatbot-china-melody

[93] How To Build Bots for Messenger. Retrieved from https://developers.facebook.com/blog/post/2016/04/12/bots-for-messenger/

[94] Bot Shop. Retrieved from https://bots.kik.com/#/

[95] Bot Users. Retrieved from https://api.slack.com/bot-users

[96] What are Skype bots and how do I add them as contacts? Retrieved from https://support.skype.com/en/faq/FA34646/what-are-skype-bots-and-how-do-i-add-them-as-contacts

[97] How WeChat bots are running amok. (2017, January 18). Retrieved from https://venturebeat.com/2017/01/18/how-wechat-bots-are-running-amok/

[98] LINE Developers. Retrieved from https://developers.line.me/bot-api/overview

[99] Telegram Bot Platform. (2015, June 24). Retrieved from https://telegram.org/blog/bot-revolution

[100] Constine, J., & Perez, S. (2016, September 12). Facebook Messenger now allows payments in its 30,000 chat bots. Retrieved from https://techcrunch.com/2016/09/12/messenger-bot-payments/

[101] Matney, L. (2016, August 03). Kik users have exchanged over 1.8 billion messages with the platform's 20,000 chatbots. Retrieved from https://techcrunch.com/2016/08/03/kik-users-have-exchanged-over-1-8-billion-messages-with-the-platforms-20000-chatbots/

[102] Kuligowska, K. (2015). Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents. Professionals Center for Business Research, 2(02), 1-16. doi:10.18483/pcbr.22

[103] Vugt, H. C., Bailenson, J. N., Hoorn, J. F., & Konijn, E. A. (2010). Effects of facial similarity on user responses to embodied agents. ACM Transactions on Computer-Human Interaction, 17(2), 1-27. doi:10.1145/1746259.1746261