Machine Learning (2023) 112:4789–4836
https://doi.org/10.1007/s10994-023-06383-0

Machine learning from casual conversation

Awrad E. Mohammed Ali[1] · **Avelino J. Gonzalez**[1]

Received: 13 November 2019 / Revised: 6 February 2022 / Accepted: 1 August 2023 /
Published online: 27 October 2023

Abstract
Human social learning is an effective process that has inspired many existing machine learning approaches, such as *learning from observation* and *learning by demonstration*. In this paper, we introduce another form of social learning, *learning from a casual conversation* or LCC a machine learning approach in which an artificially intelligent agent learns new information through an extended natural language dialog with a human. Our system enables the agent to add or change information in its knowledge base as a result of the human's conversational text inputs. LCC seeks to close an important gap in the state of the art that has focused on teaching computer agents how to perform specific tasks. Furthermore, LCC could also provide an efficient way to enhance the knowledge base of certain types of systems without requiring the involvement of a programmer. LCC does not require the user to enter specific information; instead, the user can converse naturally with the agent. As part of its learning process, LCC identifies the text inputs from the conversing human that contain information worth learning, and then determines whether the inputs are heretofore unknown and learns it; in agreement with what it already "knows" and ignores it; or in conflict with what it "knows" and it must resolve the conflict. LCC's architecture consists of multiple sub-systems combined to perform the above tasks. Its learning component can add new information to the knowledge base, confirm existing information, and/or update existing information found to be related to the user input. The LCC system functionality was rigorously assessed with test statements comprising various difficulty levels. Furthermore, its acceptance by human users was evaluated by two separate groups of human test subjects—one group who interacted with the system, and a second group that evaluated the logs of the interactions of the first group. The collected results were all found to be acceptable and within the range of our expectations.

Keywords Machine learning · Learning from conversation · Natural language processing · Chatbot · Classification · Memory update

Editor: Derek Greene.

Awrad E. Mohammed Ali
awrad.emad@knights.ucf.edu

Avelino J. Gonzalez
avelino.gonzalez@knights.ucf.edu

[1] Computer Science Department, University of Central Florida, Orlando, USA

1 Introduction

As humans, we are able to learn new things in the process of interacting with each other—sometimes to purposely learn something; other times, the learning happens casually or even accidentally as part of a conversation, such as for example, the score in a particular football game or where the other person had dinner last night. Such learning is particularly common in conversations among two or more people. However, if what we hear during such conversation conflicts with what we already know, we process the new information by either trying to convince the interlocutor of the mistake in his/her spoken information, or we accept the new information if we become convinced that it is in fact correct. Moreover, we often learn how to do things by observing others perform a task or exhibit some behavior.

Learning from such interaction with humans as a form of machine learning has been taken up by a segment of the Artificial Intelligence (AI) research community. One major advantage to this form of machine learning is that it uses an often-available and mostly reliable resource to train a computer agent directly—the human! The reliability and availability of the human source will vary depending on the task involved and whether or not scarce human expertise is required for the task. These machine learning approaches generally seek to create agents that mimic the actions of a human when performing a given task. These approaches to machine learning have not been as extensively investigated as have supervised, unsupervised or reinforcement learning (although there is some overlap), but several interesting and potentially transformative approaches have emerged in the last several years, such as *Learning from Observation* (LfO), *Learning from Demonstration* (LfD), *Learning from Instruction* (LfI) and *Behavioral Cloning*.[1].

Research on learning from direct human interaction has focused on learning two types of information (Anderson, 1996): (1) procedural—how to perform a task, and (2) declarative—factual information, such as "The sky is blue." Our research explores the latter—learning declarative information from direct human interaction through a dialog (spoken or via written text). In spite of the label's potential ambiguity, we refer to a repository of such information as the *knowledge base*, as other popular labels seem somehow inadequate. More specifically, we explore how a computer agent can learn such information through a natural and casual conversation with a human, and how such an agent can reason about the information provided by the human to update its own knowledge base (KB) accordingly. We refer to our approach as *Learning from Casual Conversation* (LCC). Section 2 discusses casual conversations at length.

As a point of comparison, LfO and LfD involve the use of "actors" to perform a task and record a time-stamped set of data related to the actor's actions. These data are considered to be the observations, and the data sets are called *traces*. Such traces are then used to train an agent on how to autonomously perform the same actions or behaviors performed by the human actor. There is generally no real-time dialog interaction between the system and the actor, as the learning is typically done offline after the traces have been collected. For these reasons, such learning from human interaction is not relevant to our work described here, and we thus omit a discussion of these otherwise very interesting areas of research.

[1] Behavioral cloning is a type of imitation learning where the agent receives the states and actions of an expert demonstrator as training data then the learning agent uses a supervised machine learning approach such as a classifier to replicate the demonstrator policy (Torabi et al., 2018) It is very similar in nature to LfO and LfD.

The type of learning from human interaction that is most similar to our approach is Learning from Instruction, as it involves human dialog in the learning process. LfI depends on a lesson-based learning protocol instead of one that uses examples (Goldwasser & Roth, 2011). LfI allows the user to use instructions in the form of natural language statements to teach a computer agent how to perform a task. The state of the art in LfI is discussed in Sect. 3 below.

Before describing our work, a discussion of what a casual conversation is and what LCC seeks to learn from them follows next.

2 What is a casual conversation and what does LCC seek to learn from one?

We begin with a discussion about conversation in general.

2.1 Conversations as means for information exchange

As social beings, we humans spend much time talking with each other. Prior to the development of a written language, it was the only way to exchange information. While much of our exchange of information now takes place via written language and/or via images (a picture is indeed worth a thousand words), we still rely heavily on oral expression for communication. Eggins and Slade (2004) refer to a conversation as "*...a semantic activity, a process of making meanings*" (pg. 6). Certainly, formal means of oral information exchange, such as speeches, lectures, storytelling and narrated presentations, among others, also involve "making meanings". These, however, represent a unidirectional flow of information—from the speaker to the listener(s). A conversation, on the other hand, is a contributory process in which two or more people participate by taking turns at speaking. Eggins and Slade describe two general types of conversation—one is functionally motivated (called task-based), where we "*...interact ...in order to accomplish a wide range of tasks*" (Pg. 6), while a second one is "*...talk simply for the sake of talking itself*" (pg. 6). They call the second one casual conversation. They define it functionally, albeit negatively, as "*...talk which is NOT motivated by any clear pragmatic purpose.*" More about casual conversations in Sect. 2.2 below.

The concept of turn-taking is an essential feature of a conversation. Sacks et al. (1978) call the contributions by each individual participant *Turn Constructional Units* (TCU), which are composed of a grammatically proper sentence or phrase, or of sequences of sentences and/or phrases. The end of a TCU marks a point of potential speaker change, at least ideally—assuming no interruptions. If there are more than two participants in a conversation, who takes on the speaker role next, at the end of a TCU, is negotiated, either subtly or explicitly by the participants themselves. The current speaker can determine the next speaker by directing a question to him/her. Alternatively, the most assertive participant, or the one with the most relevant follow-up TCU takes the turn at being the next speaker. Of course, there is much more to turn-taking in a conversation, but any further discussion would digress from the objectives of this paper. Those interested in further information should see Sacks et al. (1978) and Eggins and Slade (2004).

Clark and Schaefer (1989) discuss the concept of acknowledgement of the speaker's utterance (they assume spoken dialog, which we do not use in our work). That is, that for a conversation to not diverge, the listener must in some way signal to the speaker that her/

his contribution to the conversation (i.e., the utterance) has been heard and understood as the latter intended before the conversation can continue. This can be done in various ways, whose discussion is also tangential to our subject here. Nevertheless, if the listener does not understand (and realizes it), she/he can ask for clarification (i.e., "pardon?", "say again", "huh?"). On the other hand, the speaker may realize that the listener did not truly understand her/his prior contribution as a result of her/his response to it, and may attempt to repair the situation (i.e., "You misunderstood. I didn't say I was sleeping. I said I was leaving."). The LCC prototype system has the (somewhat primitive) ability to ask for clarification about a statement contributed by the human with whom it is conversing. It can do so by asking him/her to repeat it. However, this feature was not used in our testing, as the medium we used for these tests was typed text, whereas this feature is much more applicable to speech where misunderstandings are more common.

2.2 Casual conversations and their analysis

Casual conversations have been a subject of research by sociologists, philosophers, linguists and psychologists for many years as part of the larger conversation analysis field. Their work has generally focused on how to analyze casual conversations. Such a diverse group of researchers has made for diverse approaches to such analysis. Eggins and Slade (2004) provide an excellent historical discussion of the different lines of research on casual conversation, beginning with the work of Garfinkel (1967) and of Sacks et al. (1978) among others.

Gilmartin et al. (2018) state that "*Much daily talk does not seem to contribute to a clear short-term task, but builds and maintains social bonds, and is described as 'interactional', social, or casual conversation.*" (P. 51). As did Eggins and Slade before them, Gilmartin et al. clearly distinguish casual conversation from the more common task-based conversation, whose objective, structure, duration and the rights of participants to speak are not always set by the conversants themselves. They state that task-based conversations are often easier to analyze because they have a specific goal defined (e.g., order a meal at a restaurant, finalize a budget, revise a schedule), and everyone is aware of the goal. The conversation ends when the goal is achieved. Furthermore, the roles of the participants often dictate the turn-taking, especially when the social distances and/or places in an organizational hierarchy differ among participants, as is often the case for such conversations in business settings.

One of the earliest works on analyzing casual conversations was by Ventola (1979). In it she states that casual conversations are very important for "*...establishing and maintaining contact between people*" (pg. 267). She also describes these contacts as "*...social relationships with others*" (pg. 267). Ventola states that casual conversations typically happen during casual encounters, such as meeting at a bus stop, or on an elevator. In fact, Eggins and Slade (2004) based their research on extensive transcriptions of casual conversations that took place among co-workers during workplace coffee breaks. Ventola decomposes a casual conversation into various components, some of which are obligatory and others of which are optional, depending on the type of casual conversation one is having. The components, in rough sequential order are *Greeting* (G)—self-explanatory, and could be formal or informal; *Address* (Ad)—defines the addressee and the social distance between the speaker and the addressee (it is optional); *Identification* (Id) where the interactants introduce themselves (also optional); *Approach* (Ap) which serves as a "*bridge to conversation*" (Feldman 1959, pg. 149, through Ventola 1979)—can be direct or indirect. These components all address the initial part of the casual conversation. The next component of the casual conversation—and the most important one from our perspective—is the Centering (C) component. In a minimal conversation (defined by Ventola) such as in "*Hi, how are you? Lousy weather, huh?*", there may not be a Centering component, and the conversation goes directly to Leave-taking (Lt), followed by Good-Bye (Gb) (as in "*OK, gotta go. See you soon.*").

The reason Centering is the most important of the components for our purposes is that according to Ventola (1979), non-minimal casual conversations can and often do deal with some subject (i.e., a theme) and contain information stated by the conversant about that theme (i.e., the rheme). In her words, "*...they are about something*" (pg. 179). It is in the Centering part of the casual conversation that such information is normally found. In her own words, "C(entering) is realized by cognitive and informative topics" (pg. 273). It is exactly such information (*the rhemes*) that is woven into a casual conversation that we seek to learn as part of this research.

Eggins and Slade also had significant impact on casual conversations and how to analyze them. In their book, Eggins and Slade (2004) make one key distinction about the contents of a casual conversation that is quite relevant to our work: the concept of chats and chunks. They define chats as highly interactive parts of a casual conversation where the TCUs are short and turn-taking happens in a rapid-fire form. While they do not say this directly, their implication is that during chats, little or no useful information is shared by the participants. Chunks, on the other hand, are those parts of the conversation where one speaker dominates the discourse for an extended period of time. These chunks, according to Eggins and Slade, are more predictably structured and move through several stages. It is during these chunks that information worth learning is more likely to be shared by the speaker. The chunks often take on a storytelling nature, and Eggins and Slade define four genres of this: *Narratives*— tell stories with tension and excitement, resulting in a crisis and culminating in some form of resolution; *Anecdotes*—Similar to Narrative except there is no resolution to the crisis, only some reaction expressed as amazement, mystery, embarrassment, etc.; Exemplum—a way to express a "*...specific message on how the world should or should not be.*" (Eggins and Slade, Pg. 237); and Recounts—used to "*...retell events and to share the speaker's appraisal of these events.*" (Eggins and Slade, Pg. 237). Our assertion is that these genres of storytelling chunks can often contain information worth learning and where LCC focuses.

As did Ventola, Eggins and Slade also analyze casual conversations by the phases of a conversation. Certainly, the chats and chunks we described above are some of those phases. However, Eggins and Slade go further by decomposing storytelling chunks into their own phases. These are: *Abstract*, *Orientation*, *Complication*, *Evaluation*, *Resolution*, *Coda 1* and *Coda 2*. Not all are present in each of the four genres described above, but any further discussion of that would also be tangential to our objectives in this paper and therefore not included here.

They also define several types of casual conversations, of which storytelling chunks are only one. The others include *Observation/Comment*, *Opinion*, *Gossip*, *Friendly-ridicule*, and of course, chat. LCC is designed to neglect the chats and focus on the chunks/Centering segments of a casual conversation where discussion of themes and related rhemes are most likely to be present. We further discuss these issues in Sect. 2.3 below.

We furthermore define casual conversation (informally) for the purposes of our work as one where the speakers have the ability to speak with a degree of freedom about content and form; that is, not limiting the speaker to what words he/she needs to use to communicate with the computer agent. For example if the person wants to talk about cars or ask something related to cars, he/she can interchangeably and seamlessly say “cars”, “autos” or “automobiles”.

2.3 What does LCC seek to learn?

We should start by noting here that we do not claim that our LCC system performs the learning in the same manner as do humans. In fact, we are certain it does not. Nevertheless, we know that human-to-human conversation is a very complex process, with implied meanings spread throughout the conversation, and aided by the context, facial expressions, tonal variations, and general body language. Yet, in spite of their complexities, conversations are one thing that we humans for the most part can seamlessly handle quite well $\cdots$ and learn from them. So, then, what exactly is LCC designed to learn?- the rhemes in a user statement.

Clark and Schaefer (1989) state the three assumptions that are common in the various models of human discourse: (1) Common ground; (2) Accumulation of common ground; and (3) Unilateral speech actions. Common ground is the knowledge or information that all participants in the conversation (should) have in common, most importantly at the outset of a conversation. Without this, conversations could quickly diverge into meaningless babble. As a (coherent) conversation progresses, the conversational partners accumulate more common ground as new information is progressively offered and shared by the conversational partners. The unilateral speech action (in the form of an assertion) is how a participant in the conversation can add to the common ground. Clark and Schaefer argue that the third assumption is by itself not sufficient to correctly model the discourse, and that there are other speech acts that can add to the common ground (e.g., questions, instructions, and arguments). However, such a discussion is beyond the scope of this paper. We make the case that our LCC mechanism serves to accumulate common ground by the virtual agent that results from assertions made by the human participant.

Traum and Hinkelman (1992) discuss the issue of accumulation of common ground in depth, although in the context of a task-oriented conversation. (They also discuss the concept of turn-taking in a task-based conversation.) Traum and Hinkelman (1992) speak of conversation acts as a specialization of the speech acts mentioned by Clark and Schaefer (1989) and many others. The authors state that “*Grounding mechanisms are essential to the progress of spoken conversation ...*” (Traum & Hinkelman, 1992). The important thing about this paper is the featured role that accumulation of common ground takes, albeit in a task-oriented type of conversation.

An important part of a casual conversation is that of a chatting statement. As discussed in the Sect. 2.2 above, Eggins and Slade (2004) define a chat segment vs. a chunk segment in a casual conversation. LCC seeks to neglect the chatting statements because of their general paucity of useful information. This is not to say that chatting statements contain no useful information at all—indeed there may be some things worth learning that are mentioned in passing in chatting statements (e.g., “Hope the weather is better when I leave tomorrow”. “Oh, you are leaving tomorrow huh?”). Nevertheless, with LCC we have concentrated on the chunk/Centering segments because useful information is more likely to be found there.

So, then what characterizes the chat statements that are to be neglected by LCC? As with many other things in our world, chatting statements are hard to define but we generally recognize one when we see or hear one. We could not find a satisfactory and formal definition of what a (neglectable) chatting statement would be in our (admittedly non-exhaustive) review of the literature. Thus, we came up with our own description, if not a formal definition:

We consider chatting statements to be those that contain salutations, goodbyes, felicitations ("Happy New Year!"), most general comments about current subjects (e.g., the weather, politics, sports, celebrity gossip, etc.) that contain a qualitative expression (e.g., "Wow! It sure is hot today!"; "That was some game last night!"), statements that contain affective expressions (e.g., "I love your shoes!"), or questions/assertions about general well-being ("How are you this morning?"), among others. Moreover, questions (as indicated by a question mark) were rightfully classified as questions and forwarded to the Q/A module, rather than to the learning module, as they would not generally contain assertions.

LCC seeks to learn information revealed by a conversant during the Centering phase of a conversation (as defined by Ventola, 1979) and/or during the chunk segments (as defined by Eggins and Slade, 2004). Such information is declarative in nature and comes in the form of direct statements that discuss some theme. The theme does not have to be previously stated in any way and can be changed by the conversant at any time by simply changing the subject, but the statement must be associated with a general theme (e.g., football, ladies' fashion, rock music, etc.). In effect, LCC is designed to attempt to learn any statement NOT deemed to be a chatting statement or a question.

We fully recognize that sometimes an input statement may contain chatting as well as information worth learning. In statements where the chatting portion precedes the information worth learning, LCC can isolate and neglect the chatting part of a statement while forwarding the information worth learning to the Learning unit. As an example, "The weather is nice, so there is a car race today", LCC should be able to extract the information that "there is a car race today" from the overall statement if the chatting expression happens in the beginning of the sentence, as it does in the example.

2.4 Use case

The relevant definitions above are inevitably relevant to use cases. We believe that the basic concepts of LCC (although not necessarily of our prototype implementation) described in this paper could be applied to many different use cases in natural language conversations between human and machine. One general class of use case relates to the task-oriented conversations mentioned above that have specific objectives, such as making doctor appointments, reserving a table at a restaurant via telephone, deciding on a budget for a project or requesting that a specific valve in a nuclear power station be opened. Alternatively, the objective of the task could be of a higher nature, such as conversations specifically designed to debrief a human, such as when carrying out explicit knowledge acquisition from an expert; consultations between a patient and a (possibly virtual) physician, maybe in a tele-health context.

Casual conversations, as we have seen above, have no specific goal, so appropriate use cases would be quite different from those applicable to task-based conversations. One example of this would be a conversation among two "people" (where one of them could be a computer agent) discussing what two (human) friends would typically discuss while enjoying each other's company over a coffee or a beer while discussing automobiles, food, the latest political news or football. In such conversations, information to be learned can be expressed explicitly ("Ferraris are built in Maranello, Italy") or implicitly ("in my visit to the Ferrari factory in Maranello, I saw the F40 in the Ferrari museum"—the LCC could learn that the speaker has been to Maranello, Italy and that Ferraris are built there).

Another example of this type of conversation would be an interaction between a person and her/his intelligent assistant (e.g., SIRI, Alexa, Google) so that the latter may learn about the preferences of its "master" through such interaction (leaving aside for the moment some potentially serious privacy implications). It is this type of casual conversation that we specifically target with our research described here.

We particularly envision a potential commercial application to companion robots or avatars designed to keep company to a home-bound individual—elderly, sick, and/or disabled person—who may need a conversation partner to stave off loneliness and/or remind him/her of medical appointments and medicine intake schedule. We would describe these conversations between patient and robot/avatar as casual, such as between two friends, but could also contain some task-based elements in the conversations. Some small talk (i.e., chatting statements) will likely be involved, but inevitably, information worth learning by the agent may be offered by the human, either explicitly or implicitly as part of the conversation. Such casual conversations could also include questions by the human, which the system needs to be able to answer i.e., "At what time do I take medicine X?". Furthermore, if the patient is monitored by a health care provider through this companion robot/avatar system, then it is likely that some of the information provided by the patient and learned by the system (e.g., "Man, what a headache I had this morning!") may be found to require the attention of the health care provider.

2.5 Some assumptions and justifications about casual conversations for LCC

Before continuing to the next section (Work of Others), we should mention that as the early exploratory research that our work represents, we made several simplifying assumptions for LCC. These are: 1. The human conversant is assumed to be always telling the truth if he/she is deemed to be "trustworthy" (explained later). 2. Questions asked can only require yes/no answers. 3. The statements presented to LCC must contain no paralinguistic sounds such as "hmmm", "ah ah", "aha", "huh?" etc., nor swear words, expletives or slang (LOL, BFF, OMG, etc.). 4. An input to LCC must be without errors that would lead to subsequent correction by the speaker in the same or in a subsequent statement (i.e., "Sorry, I meant X, and not Y"). 5. The input statements (sentence) by the human conversant must be only one (relatively) short and complete sentence and must use proper or nearly-proper grammar. 6. Typed text is used to communicate, and not speech. 7. Reference pronouns (he, she, it, they, etc.) are not (yet) handled well by LCC, so they should be avoided. 8. Facial expressions play an important role in communication between humans to indicate humor, irony etc. without actually saying it. However, LCC does not take into account such facial expressions, although we envision an ultimate system to be able to do so. 9. Humor, irony, double negatives, obvious exaggeration, facetiousness and such other conversational "tricks" are not recognized by LCC. 10. LCC only converses with one human user. 11. LCC does not attempt to process any indication of the speaker's affect in the statement, at least to the extent that affect can be captured strictly in words. This is not a big drawback, as affect recognition typically depends on facial expressions, body language, and tone, none of which can be expressed easily or well in typed text. 12. LCC does not recognize temporal issues. For example, in the statement "John returns home tomorrow"; if this is learned, it will always say tomorrow, even months from when it was originally learned.

To be sure, these simplifying assumptions do limit the usefulness of our system at this time, and will have to be lifted in any future research. However, we believe that they are appropriate in the context of early exploratory research.

LCC was not designed to be a full conversational agent. While the LCC prototype can engage in a simple conversation with the human conversant (let's call him/her the user) through a third party chatbot (described in Sect. 4.5), it does not have the ability to respond richly. The statements made by the user and how they are processed by LCC are more important than how LCC responds to the user. The responses by LCC are thus of minimal value. Like Sacks et al. (1978), we consider a conversation to be a sequence of TCUs made by the user, where each TCU is one complete sentence and is considered an input statement to LCC. The turn-taking cues are obvious: the human completes the statement (TCU) and presses return, while LCC does as computers are wont to do - it simply waits after displaying its response.

Finally, we should note that the test statements used in the tests described in Sect. 5 do not reflect the envisioned use case of a home-bound patient conversing with a companion robot or avatar. Instead, they represent a conversation about specific themes between two "friends", one of which is the LCC system. The topics chosen were food and automobiles. The main reason for this is that we did not have access to statements about conversations between patients and caregivers, as they are sensitive to privacy issues and normally regulated by privacy laws. Additionally, these conversations reveal information that is invariably time-sensitive (e.g., if the patient has a headache now, such may not be the case in 24 h). As stated in item 12 above, LCC is not designed to handle time-sensitive information at this time—that is one of our areas of future research. Nevertheless, we do not believe the type of input statements used for our tests in any way lessens the value of the assessments described in Sect. 5.

3 Related work

Liu and Mazumder (2021) emphasize the importance of learning from conversation, as the existing chatbots are trained on handcrafted and/or on limited database examples. They also suggest the importance of continual learning of skills and knowledge as they chat with users. Their work is centered on extracting information from the user input that can be used later for conversation. The authors also highlight the difficulties faced during learning in real time and how to revise incorrect knowledge previously learned by their chatbot. Their paper does not go into great detail on how the learning is being performed or how information is matched to what the chatting bot already knew. It does not include any test results or even examples of conversations; yet, it serves to indicate that other researchers are addressing the same problem as our research being reported in this paper, and identifying the difficulties involved in such a task.

The earliest form of LfI provided feedback to a system as an additional resource for learning. For instance, Kuhlmann et al. (2004) used instructions to provide feedback in plain English about the system's performance while using reinforcement learning in RoboCup KeepAway soccer. The system was evaluated by checking the benefit of adding different feedback messages independently as well as jointly. The results were reported by measuring the average episode duration over time. An episode represents the interval of time between when one of the two teams (called the *takers*) takes possession of the ball away from its opponent (the *keepers*). The results indicate that while the advice did not speed up the learning process, it was found to help the learner agents perform better than without it. Torrey et al. (2005) used reinforcement learning to learn how to move in a KeepAway game and later used verbal advice to transfer the learned task to play BreakAway, a very similar game. Transfer advice involves determining which actions are best in a given situation, and those actions should be related (i.e., they can be transferred to a second task that is similar to the first one, with only minor modifications). The results reflect that advantages resulting from transfer advice started to become evident after 2500 games, and it led to higher performance results than without it.

LfI requires identifying the instructions explicitly, including their post-conditions, actions and goals. One example is the work reported by Rybski et al. (2007), involving an interactive task training for a mobile robot. In this system, the learning process is a mixture of learning from demonstration and learning from instruction. The user must verbally specify when the robot needs to start learning. Therefore, learning from instruction places strong syntactical constraints on the user utterances. An example from Rybski et al. (2007) is shown below:

When I say deliver message: if person1 is present, give message to person1; otherwise, report message delivery failure; go to home.

Therefore, learning from instruction depends on the clarity of the communicated information, which could result in the system providing the wrong instructions if the communication environment is noisy. Noise in the environment can come from a noisy physical environment (e.g., in an industrial setting, in a setting with a TV or loud music blaring in the background), from noisy electronics and/or from the inherent ambiguity of natural language. Improving this process is not a trivial task (Goldwasser & Roth, 2011).

The research discussed above present interesting approaches in this relatively new area of research. Nevertheless, there are several major differences between the LCC system and learning from instruction:

- LfI is targeted toward task-oriented systems, where there is a specific task that the learning agent/robot needs to accomplish. LCC on the other hand, is designed to be an open-ended learner that acquires new information and/or modifies its existing information through a casual conversation with a human. This casual conversation has no defined objective.
- In LfI, the new task can be changed only slightly from the original task that the agent already "knows". This is because the agent's capabilities and actions are fixed. On the other hand, LCC is not limited in such a way because it is designed to learn any statement entered by the conversant that is deemed worthy to be learned, regardless of whether it is almost already known or not.
- LCC can make decisions about what should be learned as new information, whether existing information should be modified in light of the statements by the human conversant, and what statements should be neglected because the information contained in them is already known or deemed to be wrong. This means that LCC does not always accept the information put forth by the human conversant. This feature is important because it protects the system from acquiring incorrect or inappropriate information.

Other recent works in the LfI literature involve using human feedback in natural language to improve a computer agent's dialogue skills during a conversation. Weston (2016) developed an agent that can learn through dialogue with a human who provides feedback as a form of natural supervision. Weston studied the effect of using different modes of feedback on the agent's ability to learn. These modes were:

1. Imitating an expert: the learner agent tries to imitate an expert while the expert answers a human teacher's questions.
2. Positive and negative feedback: the learner agent receives feedback from the teacher to reflect on whether its answer is correct or not.
3. Answers supplied by the teacher: the teacher provides a correction to the learner agent's answer through feedback.
4. Hints provided by the teacher: hints are provided to the learner agent instead of giving the right answer directly.
5. Supporting facts provided by the teacher: the teacher provides supporting facts to show that the learner agent's answers are incorrect.
6. Partial feedback: feedback is given only 50% of the time.
7. No feedback: no feedback is given to the learner agent.
8. Mixture of imitation and feedback that combines modes 1 and 2.
9. Asking for help/corrections: the learner agent asks the teacher to provide the correct information.

The learning model for Weston's system (Weston, 2016) uses an end-to-end memory network. This memory architecture takes the last user utterance as an input, plus a set of memories relevant to the user input, which are called *contexts*. Then, it converts that input into a vector. The main task of the memory network is to generate an output based on the user input that best matches the context. There are four training strategies that can be used by the system (Weston, 2016):

1. Imitation learning that involves imitating the user's speech. The memory model here is trained using stochastic gradient descent and it does not have restrictions on what to imitate.
2. Reward-based imitation: here, the agent only imitates the action that received a positive reward, which can only be received if the action is correct. This eliminates poor choices from being part of the learning process.
3. Forward prediction: This process predicts the correct next answer for the teacher by using the answers provided by the agent to the speaker question/input.
4. Reward-based imitation plus forward prediction that combines strategies 2 and 3.

The effect of using these modes was tested using the different training strategies to improve the probability of answering questions correctly. A task is considered successfully passed if an accuracy of at least 95% was achieved. Test results reported the accuracy across the four training strategies to answer questions related to the different modes discussed above. The results indicated that in general, Reward-based imitation plus forward prediction that combines strategies 2 and 3 performed better than the other strategies.

The work of Li et al. (2016) can be considered an extension to Weston's work (Weston, 2016). It is based on asking questions to improve system performance. The authors investigated the importance of the learner agent asking questions by examining how it can benefit in two settings; (1) offline supervised learning, and (2) online reinforcement learning that also includes knowing when to ask. The authors focused on three tasks:

- Question clarification, when the learner has difficulty understanding the user and/or his/her question.

- Information operation, when the agent asks a question to clarify existing information in its KB.
- Information acquisition, when the learner agent's information is incomplete (Li et al., 2016).

Question clarifications are needed when the user input contains typos or spelling mistakes. Without this feature, the authors must ensure that spelling mistakes are not present in the testing or training data. To solve this problem, the authors suggested two methods; (1) asking the user to rephrase the question, such as responding with "what do you mean?", and (2) verify the question by relating the misspelled word to another question without spelling mistakes, and responding with the errorless question.

The system limits the agent's tendency to ask questions to make the conversation sounds natural by setting a cost associated with this action. This is accomplished by allowing the learner agent to answer the proposed questions by the human teacher directly if it knows the answers, or when the system thinks that the question is so difficult that no clarification could help. In the latter case, the system responds by indicating that it has no answer for such question. The system also depends on user feedback (both negative and positive) to evaluate its performance. Evaluation was performed on both offline and online reinforcement learning where the authors used different combinations of the training strategies discussed earlier. The results indicated that asking questions in both settings was helpful and improved the accuracy of the answers provided by the system. This was because it provided additional information for the learner to answer correctly. This was confirmed by the results, where the agent achieved an accuracy of less than 1% if it did not ask questions.

Zhang et al. (2017) presented a system that enabled the agent to learn the English language by interacting with a teacher and consider his/her feedback. The focus of their work was on engaging the agent in the learning process. This process is influenced by the idea that imitating the speaker is not enough to drive a successful conversation. The learning process involved two components:

- Imitation: learning a language model by observing the teacher's behavior during a conversation. The training data depend only on the teacher's utterances, while the training process depends on predicting future words and sentences.
- Reinforcement: learning involves trial and error using the teacher's feedback.

Interaction with the learner agent involves asking questions of it, which the learner agent answers, or describing an object and the learner agent repeats the description. If the description is correct, the learner agent receives a reward in the form of positive feedback.

Evaluation involved asking the agent to answer questions considering the case with four different objects surrounding the agent in each direction (S, N, E, W). The teacher then interacts with the agent in three different forms: (1) asking a question as "what is on the south"; (2) describing objects as "apple is in the east" and the agent needs to repeat the statement; (3) teacher saying nothing, then the agent describes objects around it and gets feedback from the teacher. The agent receives a positive reward of + 1 if it answers correctly or produces correct statements, and a negative reward of – 1 otherwise. Test results showed that combining imitation with reinforcement learning achieved better results—approximately 20% improvement in accuracy compared with applying each one separately.

In the previously described systems (Weston, Li et al. and Zhang et al), the authors defined specific tasks, such as describing an object. Later, the authors tried to measure the system performance with and without interactions with the human. On the other hand, LCC seeks to have a computer agent learn declarative information and differentiate between what is considered as important information worth learning and what is not considered important. LCC shares with the above systems the idea of using conversation to improve the system's performance, but the objective is quite different because LCC focuses on learning information rather than learning how to perform specific actions.

4 Approach to LCC

LCC is composed of two major stages (units). First, LCC determines whether the utterance (or input text as in our case) made by the human user includes information worth learning. Utterances that probably do not contain learnable information include small talk such as "Wow, it is really hot today!" (we refer to these as *chatting statements*), and questions ("What is the temperature now?"). On the other hand, an example of information worth learning could be "The temperature today broke the record high of 99° F"). As we indicated in Sect. 2, we do acknowledge that chatting statements often include information worth learning; however, extracting that information is left for future research.

We refer to the first unit as the *classification unit*, as the system seeks to classify the human's input as either *chatting* (disregarded in the context of learning), *question answering* (e.g., an answer is to be provided from what is currently in the knowledge base), or *information worth learning*. If the input is classified as chatting, the chatbot agent (see Sect. 4.5 below) responds with a neutral, yet generally relevant response (e.g., "yes, it sure is hot"), but the input is otherwise ignored from the standpoint of learning. If classified as a question, then the agent should use its knowledge base to answer the question. However, if classified as information worth learning, the input statement is passed to the learning unit for further processing, which is the second major stage of LCC.

We should note here that one of the innovative aspects of the LCC system is that the knowledge base is a lexically organized semantic network. That is, the nodes in the network contain declarative sentences in natural language that convey information (e.g., "The sun rises in the east"), linked by arcs that indicate the degree of similarity (discussed in detail later) with neighboring nodes (that also contain sentences). The learning stage searches the network for the most similar sentence to the human's text input or utterance, and acts in accordance to how similar it is to the most similar sentence found. If sufficiently similar, the learning stage assumes it already knows the information and does nothing; if in conflict (e.g., "The sun rises in the west"), then it disputes the human's input, but may accept it as an update under some circumstances; if not sufficiently similar, it assumes it is new information and adds it to its network. In the last case, it must re-arrange the network to suitably position the new sentence among its most similar neighbors and compute their quantitative similarities. These stages are discussed in greater detail in their respective sections below.

The main components of the LCC architecture, therefore, are the classifier unit, the knowledge base unit and the learning unit. However, the LCC system contains other auxiliary components (i.e., units) as shown in Fig. 1. Each unit consists of multiple subcomponents that operate using different algorithms, as briefly described below. LCC component units can be summarized as follows:

- *Input unit:* accepts the user input statement in the form of unrestricted text.

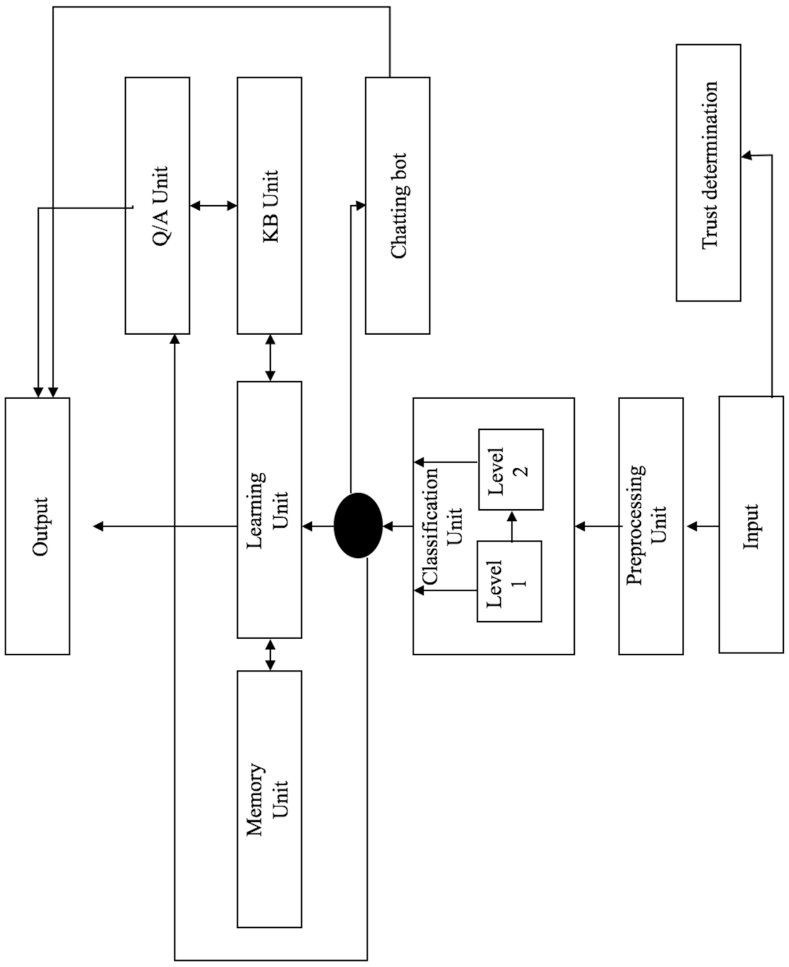


Fig. 1 LCC architecture

- *Trust determination unit:* determines the user's trustworthiness to allow her/him access to the learning feature of LCC. This includes the user's credibility on the topic being discussed as well as his/her intentions for engaging in a dialog with LCC.
- *Preprocessing unit:* applies text preprocessing to the user's input statements.
- *Classification unit:* a major unit that determines to which unit the user's input will be directed next—chatting, question/answering or learning.
- *Knowledge base unit:* a major unit that contains the agent's information.
- *Learning unit:* the third major unit that is responsible for learning the information presented when it is deemed by the classification unit to be worth learning.
- *Memory unit:* an auxiliary unit that consists of text files that record the users' previous interactions.
- *Question/Answering unit:* responsible for answering questions related to the knowledge base of LCC.
- *Chatting unit (also called the Chatbot):* responsible for responding to users' chatting statements.
- *Output unit:* provides the LCC output to the user.

We now describe these units in detail.

4.1 Trust determination

The trustworthiness of the human user is an important factor to consider when learning from casual conversation. In human-to-human conversation, whether we decide to learn something from the person with whom we are conversing or decide to neglect it depends on the trust we might place on the person providing the information. We consider this trust to be two-sided—trust in the intentions of the potential user, and trust that the conversant is knowledgeable about the topic of discussion, and thus is a credible source. Consequently, if the human conversant is either not credible on the subject or is judged to have dubious intentions (i.e., purposely attempt to provide lies to corrupt the system), then LCC should neglect the information she/he provides.

As can be seen from Fig. 1, the first interaction a user has with LCC is designed to determine the trustworthiness of the user. LCC authorizes humans labeled as *trustworthy* to have full access to its learning process (i.e., add and modify new information). Humans labeled as *untrustworthy* users, on the other hand, can only access the chatting and the system's question-answering units. If users deemed untrustworthy (i.e., unauthorized) attempt to provide information to LCC, the system will ignore the statements and instead either change the subject or proceed to chat with the user.

The current version of the LCC prototype merely asks the user to enter "1" if he/she is self-labeled as trustworthy and "2" if not. If the user enters "1", he/she will have full access to add and modify information in the KB, chat with the system, and ask questions. If, on the other hand, he/she enters a "2", then the user can only ask questions and chat with the system. This simple approach is a placeholder until the trustworthiness issue can be fully addressed in future research. Determination of whether or not a potential user is trustworthy, however, is an area of research all onto itself. As our work on this project focused on the classification and learning functions of LCC, we opted to leave this for future research, when we hope to explore more complex approaches to trust determination in LCC, such as password-controlled access, authority evaluation through questions, facial recognition, fingerprint recognition, internet inquiries about the user, and other such security measures.

4.2 Input and text-preprocessing

LCC accepts typed user input in natural language English without putting restrictions on how the user input should be formatted. LCC applies text pre-processing to normalize the user input to be more compatible with the information in its knowledge base as well as with its format. This process facilitates the classification of the user's input and its matching with existing information in the knowledge base. The pre-processing step involves changing upper-case letters to lower-case for consistency with what is found in the knowledge base. A spell checker from the autocorrect library in Python is used to correct words misspelled by the user.

We decided to use written text to communicate with LCC rather than spoken speech to avoid any errors that might result from automated speech recognition. As the accuracy of automated speech recognition systems continues to improve and mature, it will be our goal in future research to upgrade the LCC system to also accept spoken speech as input.

4.2.1 The classification process

This is the first of the three major units in LCC, and performs an important step in the LCC process by classifying the user input as information worth learning, as a question, or as a chatting statement. This is essential because a wrong label will result in an incorrect action by the system.

As shown in Fig. 1, LCC uses two levels of classification to determine the type (label) of an input statement presented to it. Ensemble Learning is used on each of these levels independently. Ensemble learning uses various classifiers on the same input data, then applies a majority voting scheme on the labels proposed by the different classifiers to determine the most likely label for the given statement. The benefit of using multiple classifiers is the improvement of the overall decision-making process that minimizes the chances of making incorrect classifications (Dietterich, 2002). Our implementation of Ensemble Learning uses the majority vote from a Naïve Bayes classifier (Dai et al. 2007; Naïve Bayes text classification https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html), a Decision tree classifier (Apté et al., 1994), and a Max Entropy classifier (Chieu & Ng, 2002; Nigam et al., 1999). These classifiers employ different mechanisms to determine the label of the given input. This diversity of classification mechanisms likely improves the overall decision. Moreover, these particular classifiers have been successfully used in the literature for text classification.

4.2.2 The two levels of LCC classifiers

The overall algorithm for the classifier unit is shown in Algorithm 1. The first level of the classification process (line 3) classifies the input text at the sentence level, where all

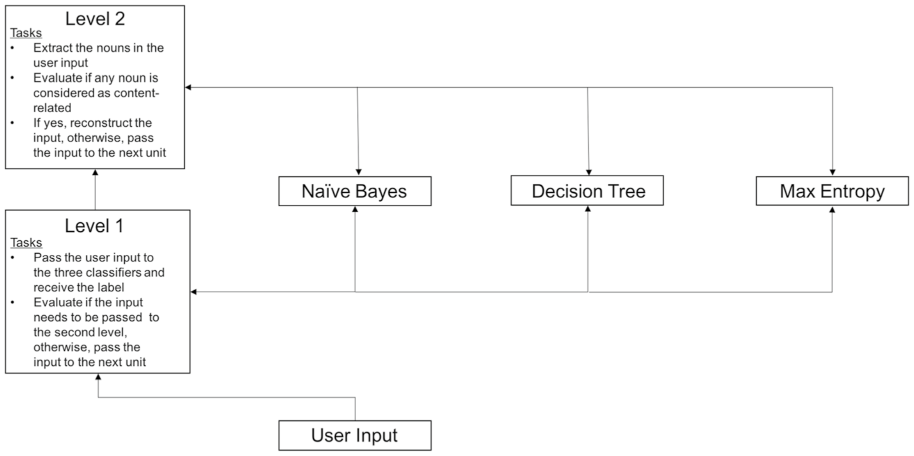

Level 2
Tasks
• Extract the nouns in the user input
• Evaluate if any noun is considered as content-related
• If yes, reconstruct the input, otherwise, pass the input to the next unit
Naïve Bayes
Decision Tree
Max Entropy
Level 1
Tasks
• Pass the user input to the three classifiers and receive the label
• Evaluate if the input needs to be passed to the second level, otherwise, pass the input to the next unit
User Input

Fig. 2 A block diagram to show level 1 and level 2 classification process

words in the user input statement contribute to the classification process. When unanimity is achieved among the three classifiers with regards to the label of the input text, the classification process ends here and the second level is bypassed altogether. LCC then forwards the labeled input to either the chatbot, the Q/A unit, or the learning unit, depending on the label determined.

However, when the decisions of the classifiers in level 1 are not unanimous, the second level is invoked (line 7 in Algorithm 1). The idea behind creating a second level is to give an opportunity to the classifiers to reconsider their decisions on the first level. This is done by classifying nouns in the sentences without considering the entire sentence. This second level uses the same three classifiers, but they use part-of-speech tagging to extract the nouns from the input text. These nouns are fed to the three classifiers to determine their labels. However, the second level does not make use of the results generated in the first level.

If there is unanimous agreement among the classifiers in the second level about the label of the user input being information worth learning (there must be at least one noun is a non-chatting statement for it to be evaluated in the second level), the sentence is reconstructed to include everything after that noun, and neglects the words coming before this first non-chatting noun in the sentence. However, this will only happen if the reconstructed sentence is a complete sentence in order to ensure that LCC doesn't consider a sentence fragment to be worth learning. To determine completeness, LCC checks whether the resulting text contains a noun and verb, and the sentence consists of at least three words.

If there is no unanimous agreement on the information worth learning label in the second level, then the user's input is labelled as a chatting statement. Our future research will apply a similar concept to other variations of sentence structures. Such an extension will consider removing chatting-related information found in different parts of a sentence. This will require rebuilding the sentence and making sure that its meaning remains accurate and grammatically correct. An overall diagram that shows the two levels and how they interact with each other is depicted in Fig. 2.

```
while input != termination-statement do
    preprocessing(input);
    level1-classifier(input);
    if majority-vote-count < 3 then
        POS(input);
        if POS(input) is noun then
            level2-classifier(nouns in input);
            if lable == content-related && majority-vote == 3 then
                input = reconstruct(input);
    if input == chatting then
        output chatbot(response);
    end
    else if input == question && label == content-related then
        QA(input);
        output QA(response);
    end
    else if label == content-related then
        learning(input);
        output learning(response);
    end
    if input == termination-statement then
        Exit();
    end
end
```

Algorithm 1: Directing the User Input to the Right Component

4.2.3 Classifier training and testing

To train the classifiers, 650 sentences from various sources in the Internet were collected. Sentences labeled as *non-chatting* were gathered from an online article about random facts (https://www.factslides.com) as well as from data extracted from reviews on various topics as part of a project at the University of Illinois (Ganesan et al., 2010). To train the classifiers to recognize chatting statements, we used statements from the chatterbot training corpus for greeting (Cox, 2017) and for general conversation such as "good morning, how are you?" and "Tell me about yourself" (Cox, 2019). Moreover, we generated 20 additional statements that were similar to those statements, such as "how was your day?" and "can we go to a movie?" A full list of those statements can be seen in appendix C in Mohammed (2019). The classifiers were trained on 80% of the collected data and tested on the remaining 20% The sentences were labeled manually by the lead researcher. We used the 80/20 split based on the Pareto Principle (Dunford et al., 2014) that is commonly used in splitting training and testing data, such as in the work of Chang et al. (2010) and Joseph and Vakayil (2021). For example, the sentence "All cars have four wheels" was labeled as information worth learning as it contains declarative information. While "Hi, the weather sure is lovely today" was labeled as a chatting sentence.

4.3 Representation of information in the knowledge base

The knowledge base contains two forms of representation of the same information: a text file that holds the content of the knowledge base as strings in plain text, and a semantic network that is constructed based on the contents of that text file. Those representations are continually updated in real time as a conversation with the user progresses.

We opted to include these two forms of representations for two reasons: (1) the text file will be more easily ported for use in other applications, such as for example, using the learned knowledge for intelligent assistant systems such as Siri®, Alexa® or Cortana®, and (2) the semantic network is used to speed up the process of finding matches in the knowledge base for a user input statement during the learning process.

In our modified version of a semantic network, the nodes represent sentences rather than words, and the edges contain the similarity scores between two neighboring (i.e., similar) sentences. This approach was chosen because it is more appropriate to relate sentences that include similar information. The edges between the nodes are weighted using the Jaccard similarity measure (https://scikit-learn.org/stable/modules/ generated/sklearn.metrics.jaccard_score.html) that is typically used to represent the similarity between two sentences/documents. A value of 1 means that the sentences are identical; a value of 0 means that they are completely dissimilar. Values between 0 and 1 reflect the degree of similarity. Therefore, sentences with similar meaning are linked using the Jaccard similarity measure to represent the degree of similarity between the sentences in connected nodes.

The Jaccard index is defined as the size of the intersection of the two sentences (number of similar words) divided by the size of the union of them (number of unique words in both sentences).

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

Where A represents the set of words in the first sentence and B holds the set of words in the second sentence.

In order to calculate the similarity between two sentences, LCC first applies lemmatization to return the words to their roots; this facilitates the word comparisons. For example, to measure the Jaccard similarity between "this girl has a pretty dress" and "those girls have pretty outfits". By applying lemmatization, the words "this" and "those" become "this", "girl" and "girls" become "girl", and "has" and "have" become "has".

The generated semantic network uses undirected edges to connect nodes whose Jaccard similarity measure is above 0.1. Therefore, the generated networks are typically sparse.

The benefit of transforming the text file into a semantic network is to take advantage of such sparseness to prune away the low likelihood search space for the learning process. This speeds up matching the user input statement with the existing information in the knowledge base, as it precludes the need for an exhaustive search of the knowledge base. For example, assume the user input contains information about cars such as "my car has four wheels". When LCC compares this input with the first sentence in the knowledge base, say "the sky is blue", there is no similarity between the two sentences. Therefore, LCC will exclude all the neighbor nodes of "the sky is blue" from further comparison, and move on to compare the input to other nodes. This is further discussed in Sect. 4.6 below.

There are two ways to modify the knowledge base: (1) by adding a new piece of information and (2) by modifying existing information. Adding new information to the text document version of the knowledge base is accomplished by simply appending the new sentence as a string to the end of the text file. The semantic network can then be reconstructed from the updated text file to add the new node and place it in its proper location. This will allow the new node to build connections with the existing nodes. The new node is assigned a new unique numerical label.

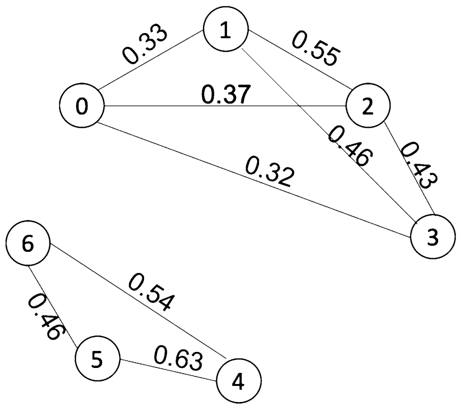


Fig. 3 Example of the semantic network

Fig. 4 The sentences for the nodes in Fig. 3

0: cars have 4 wheels.
1: about 165,000 cars are produced every day.
2: it would take less than 6 months to get to the Moon by car at 60mph.
3: the first car accident occurred in 1891, in Ohio.
4: bananas are slightly radioactive.
5: bananas don't grow on trees.
6: bananas are the most popular fruits in the US.

Modifying an existing piece of information requires updating the knowledge base by replacing an existing sentence with the user input statement that is deemed to be valid information worth learning by the classifier unit. Therefore, the new node is placed in the position of the previous node whose information is to be overwritten, and receives the node ID of the overwritten node. That will trigger a rebuild of the semantic network, as the new node might have different connections with existing nodes than had the node being replaced.

The knowledge base (both, the text file and the semantic network) is updated repeatedly to ensure that new changes in the knowledge base are always reflected in the semantic network. This is done immediately after any and all changes made to the text file.

Because the network uses ID numbers (integers) to represent the sentences in the knowledge base, a relational database is created to map those ID numbers to their original sentences. This database provides an efficient way to map the ID numbers back to their respective sentences and is updated after every change is made to the knowledge base during the learning process. Examples of the generated semantic networks are shown in Fig. 3, and the dictionary mapping between the nodes in these networks and their IDs is shown in Fig. 4.

It is worth mentioning here that there are other potential approaches for representing vectors of words. Therefore, it is appropriate at this point to briefly discuss some of these methods and point out how they differ from our implementation.

Vector of words representation e.g., Word2Vec (Goldberg & Levy, 2014) represent words as multidimensional continuous floating-point numbers in which semantically similar words are mapped to nearby points in geometric space. Each point captures a dimension of the word's meaning to humans, as the LCC semantic network does not reflect the meaning of a word and semantically similar words have similar vectors. This means that words such as "motor" and "engine" should have similar word vectors to the word "car" because of the similarity of their meanings. LCC does not represent words as vectors: instead it uses Jaccard similarity to cluster/link together sentences with similar meaning in the knowledge base. In other words, LCC doesn't per se use a vector representation for the data in the knowledge base; rather, each sentence has a unique number that links semantically similar information together. As a result of the runtime efficiency tests run as part of this research, (see Sect. 5 below) LCC clearly seems, at this point in its development, to be better suited to small and medium-sized knowledge bases. The long runtimes it takes to search and retrieve information from large KB's results in an unacceptable user experience. Our future research will explore other means to represent, store and index the information that can result in significantly faster responses to user interactions. Using word2vec representation could be beneficial to the LCC when considering large knowledge bases.

Another representation that could prove beneficial is that of Bidirectional Encoder Representations from Transformers (BERT). The BERT system is a set of pre-trained language models that can be fine-tuned for downstream tasks (Devlin et al., 2018). It is designed to help computers understand the meaning of ambiguous language from the surrounding text. It is based on transformers (Vaswani et al., 2017)—a deep learning approach that solves sequence-to-sequence tasks taking in consideration long-range dependencies between the words. Similar to other deep neural networks, it uses encoders and decoders; beyond that, however, transformers use a mechanism called self-attention (Luong et al., 2015) that assigns different weights based on the significance and importance of each part in the input data. Self-attention helps passes the position of the word to the decoder in a way similar to how humans read a text, as we always focus on the current word, but we remember keywords related to what we had read before to understand the whole sequence. Transformers have been widely used in natural language processing for tasks such as summarization and language translations. For example, to figure out the meaning of a word, it is important to know the context in which that word appear and the further we go, the more accurate the prediction for the meaning will be. BERT is different from other deep learning approaches because it is designed to read both, the word before and the word after at the same time. The main difference between systems such as BERT and how LCC represents the information as semantic networks is that LCC doesn't use a pre-trained model on a large volume of training data; rather, LCC makes use of a small to medium size volume of training data to train its classifiers to classify the user input. Yet, using a pre-trained model such as BERT could improve the classification process for LCC. We will consider using it in our future research.

4.4 Memory function

The memory function unit saves prior information about the user and his/her statements made to LCC. The memory can be accessed only through the learning unit, as indicated in Fig. 1. LCC memory consists of three different dictionaries: (1) one in which LCC saves information related to a user's trustworthiness, (2) a second one where logs of prior interactions are kept, and (3) a third one that holds information related to earlier modifications of the knowledge base.

In general, a memory model is closely related to learning because it is necessary to have somewhere to keep records of the input information prior to learning it, as well as to remember the users who have previously interacted with the system. The memory unit

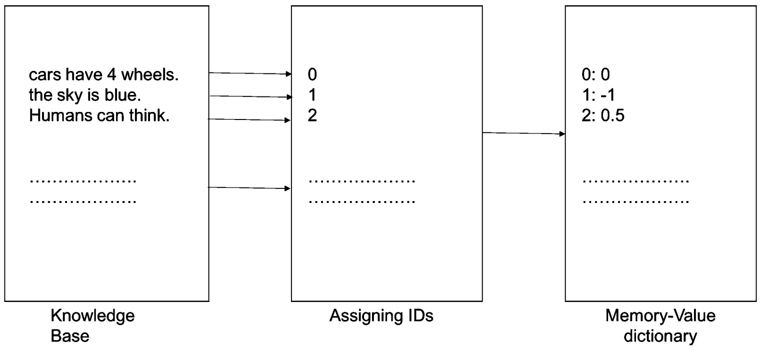


Fig. 5 Memory representation of the numerical associated values

saves previous modifications in the knowledge base to remind returning users what modifications had been made earlier as a result of their prior interactions with LCC. It can make the conversation sound more natural if recently learned knowledge can be brought into a conversation later, specifically in Q/A. Those modifications are tracked during the LCC learning process by assigning a negative integer to information that is modified, a positive integer for confirmed information and a floating point number for new information. LCC would be able to know the type of information (modified, or new) by checking its assigned number. Those numbers are assigned by the learning process to determine the state of the information. Inputs that contain chatting statements or questions are not considered by the memory unit, as it is not important to remember them. More details about this are included in Sect. 4.6.5. An example is shown in Fig. 5. For the dictionary that records the modification of previous information, the key is the node ID, and its associated value is a list of previous versions of the original information that was in the knowledge base. It is important to note that this dictionary only holds information about modified sentences. We did this to make the size of the document more manageable.

For the user's previous interactions file, the key is the username, and its associated value is a set of his/her input statements from prior sessions with LCC. The updated information is saved in a text file corresponding to that user when the user ends his/her interaction with the system.

4.5 Chatbot

LCC uses a chatbot to produce responses for statements classified as chatting, or for when the user enters information worth learning information but he/she has been classified as an unauthorized user. However, as we have stated earlier, LCC was not intended or designed to be a full conversational agent. The chatbot is built with the ChatterBot library (ChatterBot-machine learning, conversational dialog engine, https://chatterbot.readthedocs.io/en/stable/) that uses searching and classification algorithms to produce a variety of answers. ChatterBot uses an example-based model and the responses are selected by choosing the best matching answer to the user input. The training data are stored in the form of questions and answers. The program can also be trained from the user input, as it saves the user's input and the statements generated by the system as responses.

ChatterBot consists of four adapters, (1) input adapter that accepts the user input and passes it to the chatbot; (2) storage adapter that stores the information in a database; (3) logic adapters that are responsible for selecting the answers; (4) output adapter that returns the answer.

ChatterBot has multiple logic adapters that use Naïve Bayes classifiers to determine whether a given input by the user meets some criteria to guarantee a response by the system. It also uses search algorithms to help the system choose a response by measuring the similarity of an existing statement to the input statement, and the frequency of having similar responses that have occurred before.

The logic adapter plays the role of dialogue manager; therefore, it is responsible for finding an answer to the user input. The database in Chatterbot is divided into categories: For example, computers, food, greetings, health, etc. An example of how the knowledge base for the Chatterbot database is organized for the computer topic is shown in Fig. 6. The logic adapter first searches the database for a known statement that is similar or close to the user input. Later, it selects a known response to that statement, where there are usually several of them. For instance, if the user asks "what is a computer?", the logic adapter will first search the knowledge base to find the best match, which in this case will be an exact match to the first question shown in Fig. 6. Later, the logic adapter will respond to the user input with one of the responses on the left that will be chosen at random. More information can be obtained from the Chatterbot documentation (ChatterBot-machine learning, conversational dialog engine, https://chatterbot.readthedocs.io/en/stable/).

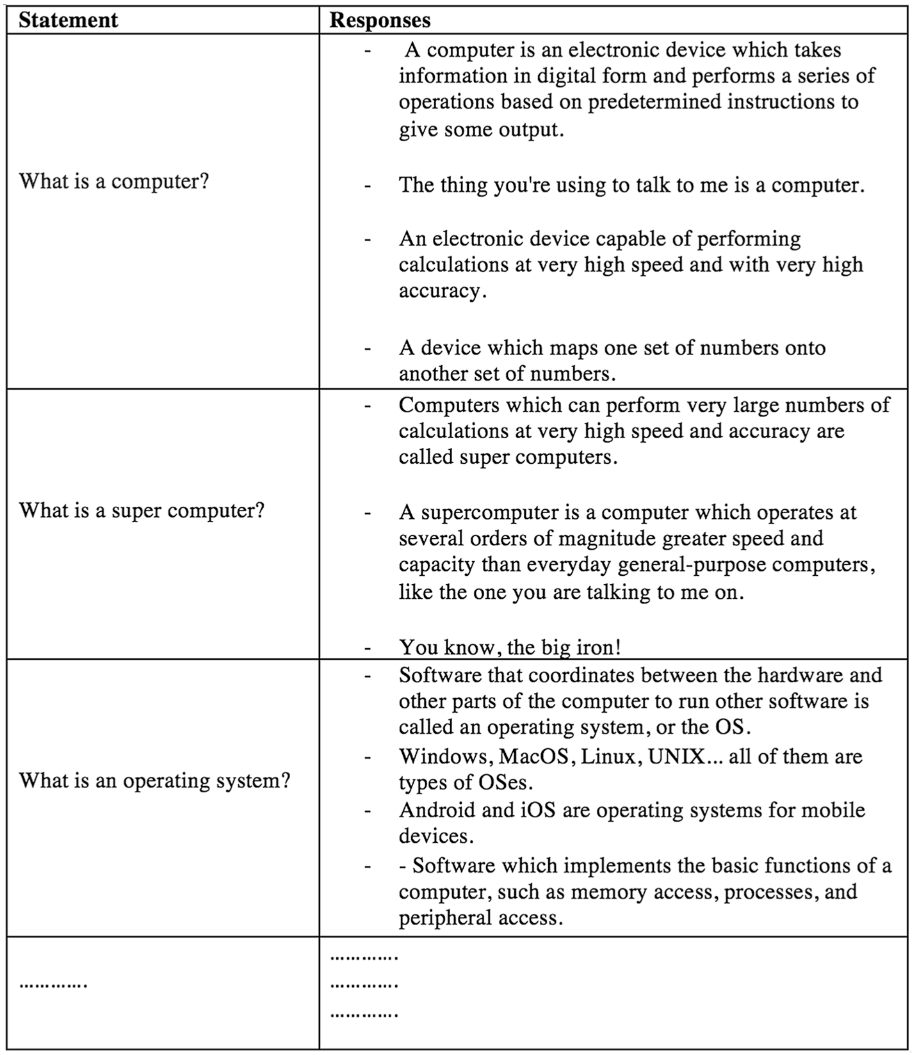
4.6 Learning unit and learning algorithm

The process of how to update a computer agent's knowledge base in real time while in a conversation with a human takes place in the *learning unit*. Therefore, the main task of the learning process is to find a correlation between a user input statement that is deemed by the classifier to be information worth learning and its best match in the knowledge base. LCC searches through all the matches made through three different similarity measures (discussed in this section below) to find the best match. How these three measures are combined in this matching process is also described in detail later in this section.

Character-based similarity measures the distance between two strings of characters based on the length of contiguous chains of characters that are present in both strings. In this research, we applied fuzzy string matching that uses a Levenshtein distance to compute the similarity between the texts strings (Yujian & Bo, 2007).

Knowledge-based similarity is a semantic measure that calculates the degree of similarity between two words using various pieces of information derived from publicly available sources, such as WordNet. Similarity measure based on WordNet, and Word alignment matching can be considered examples of this type of measure. The idea behind using different types of similarity measures was to overcome the limitations inherent in each measure, as well as to create a collaborative environment that narrows down the search space in the knowledge base to find the best match for the user input. LCC does not understand the meaning of the sentences as do humans; rather, it depends on those similarity measures to determine the closeness between two sentences.

We now discuss these measures in greater detail. However, we should note here that the three similarity measures discussed in the next three sub-sections are only used in the search and match process for the most similar node to the user input. These similarity



Statement	Responses
What is a computer?	- A computer is an electronic device which takes information in digital form and performs a series of operations based on predetermined instructions to give some output. - The thing you're using to talk to me is a computer. - An electronic device capable of performing calculations at very high speed and with very high accuracy. - A device which maps one set of numbers onto another set of numbers.
What is a super computer?	- Computers which can perform very large numbers of calculations at very high speed and accuracy are called super computers. - A supercomputer is a computer which operates at several orders of magnitude greater speed and capacity than everyday general-purpose computers, like the one you are talking to me on. - You know, the big iron!
What is an operating system?	- Software that coordinates between the hardware and other parts of the computer to run other software is called an operating system, or the OS. - Windows, MacOS, Linux, UNIX... all of them are types of OSes. - Android and iOS are operating systems for mobile devices. - - Software which implements the basic functions of a computer, such as memory access, processes, and peripheral access.
…………	………… ………… …………

Fig. 6 An example of how the database in chatterbot is organized

measures are not used to link similar nodes in the semantic network version of the knowledge base, as the latter is built using the Jaccard similarity measure as discussed earlier.

4.6.1 Fuzzy string matching

Fuzzy string matching is used as a first step to prune the search space. The pruning process is discussed in Sect. 4.6.4 below, but a look at the similarity metrics used is warranted prior to that discussion.

The fuzzy string matching similarity metric uses Levenshtein distance (Chacón et al., 2014) to calculate the similarity between two strings/sentences. There are three different variations of this measure: (a) *token sort ratio*, (b) *simple ratio*, and (c) *partial ratio*. Token sort ratio sorts the words in the strings to be compared by alphabetical order. Therefore, it does not consider the position of the words in the sentence, as it sorts the words in the strings alphabetically and applies fuzzy string matching to them. The simple ratio compares the similarity between two strings having the same length. The partial ratio can compare two strings of different lengths by finding a substring of the shorter string in the longer string. For example, "I have an appointment tomorrow" and "I have an appointment tomorrow with the doctor at 3:00 pm" have a partial ratio of 100% because the first sentence is a substring of the second.

In our approach, we used token sort ratio (fuzzywuzzy, https://github.com/seatgeek/fuzzywuzzy) to measure the string similarity rather than simple ratio or partial ratio. This was done because simple ratio requires both strings to be of the same length, which is impractical for the purpose of LCC. The partial ratio looks to find substrings rather than matching the whole string. Token sort ratio has a drawback of not considering the position of the words in the comparison, but it takes into account all the words in both strings. For instance, in the above example comparing "I have an appointment tomorrow" and "I have an appointment tomorrow with the doctor at 3:00 pm", partial ratio is computed to be 100%. Yet, the token sort ratio between them is 69%, which is intuitively more representative of their similarity. Moreover, the token sort ratio is more acceptable for our purposes because we are not looking to find a substring in both strings.

4.6.2 Similarity measure based on WordNet

This similarity measure is an example of a knowledge-based similarity measure. We used a modified version of the work by Mihalcea et al. (2006) that uses WordNet (https://wordnet.princeton.edu/) to measure the semantic score between two segments/sentences. The work of Mihalcea et al. is one of the first approaches that goes beyond simple word-to-word comparison for estimating the similarity using segments with more than three words. This approach uses a function of the semantic similarity at the word level to model the semantic similarity of the whole segment. This similarity measure combines the metrics of word-to-word similarity into a formula to measure the similarity between two sentences. There are multiple word-to-word semantic similarity measures suggested in Mihalcea et al. (2006). In our work we used *Pointwise Mutual Information* to use data collected by information retrieval *(PMI-IR)* to measure the degree of statistical dependency between two words. PMI-IR is measured as follows:

$$PMI\text{-}IR(w_1, w_2) = log_2 \frac{p(w_1 \& w_2)}{p(w_1) * p(w_2)} \tag{2}$$

where w_1 and w_2 are the two words between which we wish to measure the PMI-IR; $p(w1\&w2)$ represents the probability that both w_1 and w_2 appear together, while $p(w_1)$ and $p(w_2)$ represents the probability that each of those words appears separately in a sentence. More information about PMI-IR can be found in Mihalcea et al. (2006).

The complete process to measure the semantic similarity between the two strings operates as follows: for each word w in the first segment S_1, the system tries to identify the word in the second segment S_2 that has the highest semantic similarity (maxSim(w, S_2)). The same process is applied in the other direction, i.e., decide the most similar words in the first segment S_1 for words found in S_2. The results are summed and normalized over the length of each text segment. The results of the measure are later combined using a simple average

Table 1 Word similarity scores between Text1 and Text2 starting with S1

S1	S2	maxSim ($w \in S_1, S_2$)
Cup	Cup	1.0
Coffee	Coffee	1.0
Significantly		None
Better	enhance	0.125
Blood	Blood	1.0
Flow	Flow	1.0

of the measures in both directions. The average is computed because the similarity measures computed are typically not the same in both directions. This is a drawback in their work, as it is more practical to have the same value in both directions; however, the difference between the two values is generally small, and averaging the values makes the result more acceptable as it considers both directions. The final formula to calculate the semantic similarity is as follows:

$$Sim(S_1, S_2) = \frac{1}{2}\frac{\sum_{w \in \{S_1\}} maxSim(w, S_2)}{S_1\ length} + \frac{\sum_{w \in \{S_2\}} maxSim(w, S_1)}{S_2\ length} \tag{3}$$

The similarity value can be any value between 0 and 1 inclusive; where 1 means exact match and 0 indicates no match at all between the two sentences.

In their work, Mihalcea et al. added a weight to the similarity measure using TF-IDF.[2] For our research, the extra computation of calculating TF-IDF was not included, as our desire was to have a uniform distribution by giving each word the same weight and importance. This is acceptable here because we are dealing with short text input and we believe that every word should contribute equally when the similarity score between two sentences is calculated. Another reason to not use TF-IDF is that while TF-IDF is beneficial for documents that contain large paragraphs, as it emphasizes the importance of words that occur more often, the fact is that LCC only works with relatively short sentences.

WordNet can only compare the words with the same part of speech tags i.e., it compares nouns with nouns, verbs with verbs, etc. Therefore, to measure the degree of similarity between the user input and the knowledge base contents, part-of-speech tagging (POS) is used. Furthermore, WordNet can only include nouns, verbs, adjectives and adverbs, so other parts-of-speech tags (e.g., articles and pronouns) are ignored during the comparison.

To show an example of how the similarity measure is computed between two sentences, we show the results of calculating the similarity measure between: S1: "a cup of caffeinated coffee significantly improves blood flow", and S2: "a cup of coffee can enhance blood flow". The similarity measure starting with the words in the first sentence (S1) and compare it with the words in the second sentence (S2) are reported in Table 1. While the results of the similarity measure starting with the words in S2 and compare it with those in S1 are shown in Table 2.

As reported, the results were not identical in both directions, as we can see that when comparing the words "enhance" and "better" in Tables 1 and 2. This is because of the

[2] TF-IDF stands for Term Frequency-Inverse Document Frequency, a numerical measure that reflects the importance of a word in a document or corpus.

Table 2 Word similarity scores between S1 and S2 starting with S2

S2	S1	maxSim $(w \in S_2, S_1)$
Cup	Cup	1.0
Coffee	Coffee	1.0
Enhance	Better	0.143
Blood	Blood	1.0
Flow	Flow	1.0

limitation of this approach that we discussed earlier; however, the simplest way to overcome this problem is by taking the average of the values in both tables. Therefore, the total similarity score (calculated using Eq. 3) of Table 1 is 0.825 and the score of Table 2 is 0.829. By averaging the results of both directions, we get 0.827 which is the semantic similarity measure between these two sentences.

Additionally, we can see that the word "better" was not a part of the first sentence but it is the synset of the word "improve" because this measure uses the words most common synset in calculating the similarity score. Therefore, this similarity measure was able to relate them to each other. Furthermore, from both tables, we can see that this measure only considers the nouns, verbs, adverbs, and adjectives in the comparison.

4.6.3 Word alignment matching

The second similarity measure used here is adopted from the Semantic Evaluation (*SemEval 2015*) competition.[3] The basis of the SemEval competition is to measure the semantic similarity between two sentences using a scale from 0–5, where 5 indicates identical pairs and 0 means no similarity. The system used in our research, proposed by Sultan et al. (2015), was ranked 1^{st} during the SemEval competition. It operates by aligning semantically-similar words across the two sentences and it depends on extracting word similarity and extracting contextual similarity. To identify word similarity, three levels are considered as follows:

- Exact word match, where the similarity score is 1.
- A degree of similarity that matches non-identical words. In order to do that, the authors used a database called Paraphrase Database (PPDB)[4] to identify such relations. This level assigns scores between 0 and 1, which represent the degree of similarity.
- No similarity exists, which receives a value of 0.

An example of the alignment process is shown in Fig. 7 where five words are aligned between the two strings, including words that have a similar meaning, "awarded" and "received". Therefore, the number of alignments between the two strings is five.

[3] SemEval is an ongoing series of evaluations of computational semantic analysis systems. The evaluation involves exploring the natural meaning of the language. This task is not intuitive to machines as it is for humans (International workshop on semantic evaluation, 2015).

[4] http://paraphrase.org

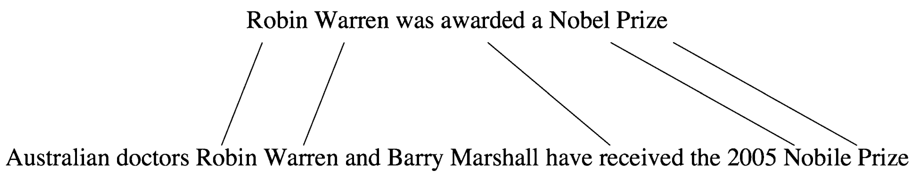

Robin Warren was awarded a Nobel Prize
Australian doctors Robin Warren and Barry Marshall have received the 2005 Nobile Prize

Fig. 7 Example of the alignment process (International workshop on semantic evaluation, 2015)

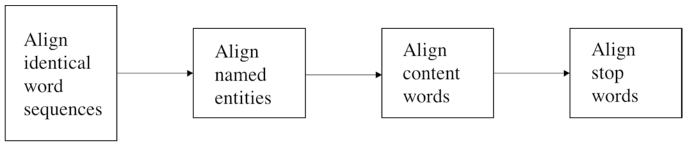


Fig. 8 Word alignment architecture (Sultan et al., 2014)

Extracting contextual similarity depends on two resources: syntactic dependencies and words occurring within a small window of the area of the two words to be aligned (three words from each side, left and right) (Sultan et al., 2014). This approach also matches misspelled words when the difference is one letter using the Levenshtein distance.

The work of Sultan et al. operates in a form of a pipeline that aligns words within the same category, as shown in their general architecture in Fig. 8. Alignment of identical word sequences is the simplest form of alignment, as it aligns sentences that have identical words in the sequence and contextually similar words in both sentences. *Named Entity* is aligned separately to allow the alignment of full and partial names such as initials and abbreviations. The Stanford Named Entity Recognizer (Finkel et al., 2005) is used to identify the names in both sentences that need to be aligned during named entity alignment. *Content Words* is used to match contextually similar words. *Stop words* use the same approach in matching contextually similar words, but because they are the last things that become aligned, it does not consider the neighboring words (on the left and right) in the alignment process, and it aligns word-to-word only.

The similarity measure is computed as follows:

$$Sim(S_1, S_2) = \frac{n_c^a(S_1) + n_c^a(S_2)}{n_c(S_1) + n_c(S_2)} \tag{4}$$

Where $n_c^a(S_i)$ and $n_c(S_i)$ are the total number of words and the number of aligned words in sentence S_i respectively. The words are aligned only when there is some semantic contextual similarity between them.

```
Learning(input, graph, value, archive)
dict = dictionary(len(graph));
create mySet;
for g in range (len(graph)) do
    if g not in mySet then
        if fuzz.token-sort-ratio(question,dict[g]) $\leq 40$ then
            for nb in graph[g] do
                if fuzz.token-sort-ratio(question,dict[nb]) $\leq 40$ then
                    mySet.add(nb)
            end
end
for g in range (len(graph)) do
    if g not in mySet then
        alignments = align(dict[g], question);
        if (len(alignments[0]) >= 2) then
            if symmetric-sentence-similarity(question, dict[g]) >= 0.5 then
                related[g] = symmetric-sentence-similarity(question, dict[g]),
                  fuzz.token-sort-ratio(question,dict[g]),len(alignments[0]) ;
end
if len(related.keys()) == 0 then
    add input to textFile;
    graph =create-net();
    update dict;
    value[g] = 0.5;
else
    maxValue = max(related, key = related.get);
    if related[maxValue][1] >= 80 and related[maxValue][0] >= 0.9) and
      (related[maxValue][2] >= 4 then
        value[maxValue] = value[maxValue] + 1.0;
        print (response of confirmation) ;
    else if related[maxValue][1] >= 50 and related[maxValue][0] >= 0.6 then
        Exchange information request ;
        if "yes" then
            change info;
            graph =create-net();
            value[maxValue] = value[maxValue] - 1.0;
            print (confirm the changing process)
        else
            print (information did not change)
        end
    end
    else
        add input to textFile;
        graph =create-net();
        update dict;
        value[g] = 0.5;
    end
end
```

Algorithm 2: LCC Learning Algorithm

4.6.4 Searching the semantic network in the knowledge base

Finding the most similar sentence in the semantic network requires searching the semantic network. Therefore, similarity measures were used not only to find the best match but also to minimize the search space, and avoid an exhaustive search of every node (sentence) in the network, as this would be computationally prohibitive given the real time requirements of the LCC application. So, the matching algorithms were not used to compare the user input with every single node/sentence in the knowledge base; rather, the user input is compared to a subset of the knowledge base that is chosen after using one of the similarity measures discussed above—the fuzzy string matching algorithm—to prune the search space. It excludes not only those nodes whose similarity to the user input is minimal, but also their neighboring nodes (i.e., those whose edges connect to these nodes).

Fuzzy string matching is the simplest and fastest algorithm for computing the similarity measure of the three similarity measures discussed above. Nevertheless, selecting an appropriate threshold to which to compare the obtained similarity scores was a critical task. An overly tight threshold would excessively prune the search space, possibly excluding good matches. On the other hand, a threshold that only prunes minimally would result in computing inefficiencies that would render the process unworkable for this real time application.

Optimal threshold determination was employed by trying different combinations of similarity scores and adjusted the thresholds until we became satisfied with how information between the KB and the user input were correlated. More details of this process can be found in Mohammed (2019). After implementing this method, we selected the value of 40%, where this value represents the threshold for deciding whether to prune a node or not. For example, the fuzzy string similarity ratio between “bananas grow on trees” and “bananas are yellow” is 40%. Using a threshold greater than 40% would mark those two sentences as unrelated; however, in reality they would still be slightly related as they both deal with bananas.

To visualize how the pruning process works, assume that the knowledge base contains information about food, cars and planets. When the user enters a statement related to cars, the learning process will start pruning the search space by computing the fuzzy string matching value between the user input and each node in the knowledge base with a threshold of 40% similarity. The 40% value provided a good compromise between pruning too much of the search space (if > 40%) and thereby possibly eliminate potentially good matches, or pruning too little (if < 40%) and increasing the runtime unnecessarily. Nodes that have scores below that threshold when compared with the user input are excluded. Therefore, all their connected neighbors nodes with sentences that deal with food and planets are likely to not be included in the search space.

Algorithm 2 describes how the process works used in LCC to prune the search space.

4.6.5 Combining similarity measures to find the best match

Algorithm 2 also shows the entire learning algorithm that combines the similarity measures described above to find the best match in the KB to the user input. The algorithm works to identify the node in the knowledge base that is most similar to the user input. Based on the similarities between the nodes in the network and the user input, and by comparing them to a predefined threshold, the information will be determined to be either a match to the user input (and therefore already “known”), information that needs to be updated, or new information to be added to the KB (i.e., learned).

Determining the thresholds to which to compare the similarity scores was a challenging task. (Note that these thresholds are different from the threshold of 40% selected for the pruning process described in the section above). Optimal thresholds determination was again employed by trying different combinations of similarity scores and adjusted the thresholds until we became satisfied with how information in the KB and the user input were correlated. More details of this process can be found in Mohammed (2019).

LCC selects the nodes that are found to be similar to the user input and adds them to a dictionary called the *related dictionary*, with their corresponding similarity scores (lines 12–18 of Algorithm 2). More specifically, the nodes that have at least two aligned words (words in common) using Sultan et al. (2015) work and their semantic similarity measure is greater than or equal to 0.5 using Mihalcea et al. (2006) method are considered to be part of the related dictionary. The threshold of 0.5 was suggested by Mihalcea et al. as a good threshold for considering two sentences semantically related. Note that the fuzzy string similarity measure is not used as criterion for inclusion in the related dictionary. However, it is used for further decision-making.

The key in the related dictionary is the node number (ID), and the values are the three similarity scores (fuzzy string matching, semantic similarity measure and the word alignment matching score) between the node and the user input statement. The related dictionary can be empty if no node in the KB is found to be above the specified thresholds.

Based on more restrictive thresholds (discussed below), LCC then determines which nodes are related to the user input by choosing the node from the related dictionary that has the highest combined similarity score. This is called the *dominant node* in the semantic network. The last phase in the learning process is when LCC determines whether the user's information (comparing it to the dominant node) is new information, a modification of existing information or similar to existing information in the knowledge base. An empty related dictionary inherently indicates that there is no similar information to the user input in the knowledge base, and that the input statement should be considered as new information to be added to the KB.

However, if the related dictionary is not empty, the decision becomes more complicated, as LCC then determines the disposition of the dominant node according to where the values fall within the following ranges:

For LCC to consider the user input as existing information, the similarity scores between the dominant node in the KB and the user input should satisfy all the following criteria (lines 26–28 in Algorithm 2):

- The fuzzy string matching score should be equal to or greater than 80.
- The semantic similarity score is more than 0.9.
- There are at least four aligned words.

The threshold of 80 means that the fuzzy string matching should match at least 80% of the information in both the user input statement and the dominant node in the knowledge base. The semantic similarity measure of 0.9 or above reflects a close semantic relation between the two compared sentences. For the fuzzy matching score, an alignment of four words was chosen to eliminate the effect of matching stop words (words that do not hold important information such as: is, are, the, etc.). However, the length of the sentences should include more than four words for this measure to be accurate.

The ranges of the matching scores were chosen to be not so tight in order to allow a small gap when a slight difference exists between the user input and the most similar information in the KB. For example, if the user enters "bananas are radioactive" and the KB contains "bananas are slightly radioactive." should be considered the same as they both refer to the same thing, which is that bananas are radioactive. During our optimal threshold determination to select the thresholds, we selected those thresholds that matched closely relevant information.

Alternatively, a fuzzy string matching score > 50 and the semantic similarity score > 0.6, LCC considers that to be a close relation to the user input, with some modifications (lines 29–38). Therefore, LCC asks the user (trustworthy, of course) if he/she wants to update information in the KB with the user text input statement. Those values are chosen not to be so high as the values that reflect exact information, and also not too low because having lower values of the fuzzy string matching and the semantic similarity measure can result in incorrectly assuming the information to be related. For example, “English is the language spoken in most US cities” and “US has many cities” are related as both refer to something in the US and its cities; however, they contain entirely different information. Therefore, they should not be considered as closely related.

Lastly, if the dominant node scores in the related dictionary are not within the previous ranges, LCC considers the information as new information and adds it to the knowledge base. Lines (40–45) add the user input as a piece of new information in the knowledge base, as none of the previous thresholds were satisfied.

During the learning process, the system assigns a numerical label to the action taken (adding, exchanging and confirming information). Those labels are assigned as identifiers so the system can know what information has been changed, added or confirmed. A 0.5 is given when the piece of information is deemed to be new information and is added to the dictionary that keeps a record of the state of the knowledge base. A label of 1 is assigned each time the user text input statement is confirmed to be existing information. A label of -1 is given when the user text input replaces what is in the KB.

4.7 Question/answering system

To answer users’ questions that are relevant to the knowledge base, LCC uses a similar approach as the learning algorithm to find the most suitable answer to the user question. However, rather than modifying the knowledge base when the user text input contains different or new information, LCC only outputs a message to the user indicating whether it has the same information or not. It replies that it has the same information when the similarity scores are within the range of confirming information (the fuzzy string matching score ≥ 80; the semantic similarity score > 0.9, and there exists at least four aligned words/expressions between the user input and the dominant node. If the sentence only includes three words, it will be ignored; therefore, anything below four will not be considered.

Occasionally, LCC will be presented with input that contains slightly different information from what it has in the knowledge base. This results in a partial match that is determined using similar thresholds to those used in the learning process with partial matching. In these cases, LCC will decide that it has related information to the user input and will put out that information. When the matching scores are not within the ranges described above, then LCC will produce a message indicating that it does not know the answer to the user’s question.

For example, if the knowledge base contains information such as “cars have four wheels”, when the user asks, “Do cars have four wheels?”, LCC will answer “Yes, I have similar information”. When the user asks, “Do cars have six wheels?”, the system answers with, “I know that cars have four wheels”. Yet, when the user asks about information that does not show any similarity to any sentence in the knowledge base, the system will simply answer, “I do not know”.

We should note that as we stated earlier, LCC can only answer a yes/no type of questions; it cannot answer questions such as "how many wheels does a car have?". We acknowledge that this is a limitation of the LCC question answering system and is left for future research.

5 Testing and results

A series of tests were performed to assess the performance of the LCC prototype system and to learn the opinions of human users about the system. The evaluation process was composed of the following three independent assessments:

- Test 1: Two functional tests were performed to test the effectiveness of the classification and of the learning units.
- Test 2: A two-stage assessment to gauge the acceptance of the LCC concept by human test subjects was carried out. The protocol involved asking the test subjects to interact with and/or evaluate the LCC prototype system.
- Test 3: A computation-only experimental assessment was done to determine the computational complexity of the LCC system. This was done to better understand the impact of increasing the size of the knowledge base on the runtime performance of the LCC prototype. The results indicate how scalable the system would be for use in real-world applications.

We discuss each of these individually.

5.1 Test 1: functional testing

The functional testing involved evaluating the ability of the LCC prototype to carry out the correct learning action (Test 1-A) and classification decision (Test 1-B) when presented with input statements of varying levels of difficulty. Test 1-B was specifically designed to assess the impact of using the two levels of decision-making in the LCC input classification process. No human test subjects were used in these functional tests. All test statements assumed a trustworthy user.

5.1.1 Functional tests, part 1-A experimental procedure: learning unit assessment with difficulty level testing

A series of tests were conducted by the authors to measure the effectiveness of the LCC learning unit. The authors composed a test set consisting of 90 input test statements/sentences that played the role of user text inputs. Each test statement was manually labeled by the authors with the expected correct action that should be taken by the LCC prototype, either: (1) add the statement as new information; (2) confirm that the statement represents existing information; (3) exchange/update existing information with the test statement; (4) answer a question posed by the input statement; or (5) respond to a chatting statement. These developer-assigned labels were considered the ground truth labels that were later used to determine the correctness of the action taken by LCC in response to each test input sentence.

The 90 test statements/sentences were placed into three groups of 30 test statements each based on their level of difficulty. The test sentences were created by the lead developer (the first author) with full knowledge of the contents of the KB, and in accordance with the difficulty criteria set forth below for each group, The grammatical structure for the sentences used in Groups 1, 2 and 3 were fundamentally the same:

$$(qualifying\ adjective)\ subject\ noun\ -\ verb\ -(qualifying\ adjective)\ object\ phrase \tag{5}$$

The differences are in what words were used for the qualifying adjectives and for the noun phrases. Our means for identifying similarity depend heavily on individual words, so if dissimilar words are used, this could result in an incorrect matching of the test input sentence with the contents of the knowledge base. We should note that the 90 sentences that comprise the three test groups were designed to achieve specific test objectives.

1. Group 1: The objective of the Group 1 test sentences was to verify that LCC had a "floor" of relatively easy sentences that it could successfully match to the KB contents. Therefore, Group 1 sentences were short sentences that used words that were largely the same as those found in the KB, although with additional qualifiers for the noun and/or the object phrase. For example, a test input of "Electric cars fake engine noise", can be matched rather easily by the LCC's similarity measures to "Most electric cars fake engine noise through speakers", which is what was contained in the KB. LCC should have been able to easily match these two sentences because "electric cars" and "fake engine noise" were found verbatim in both sentences. The differences of "Most ..." as an adjective quantifying the subject, and "...through speakers" qualifying the object would not make them dissimilar enough to result in a mismatch. Thus, LCC should respond that it has similar information in its KB.
2. Group 2: The objective of Group 2 was to challenge the LCC algorithm with input test sentences that would present a challenge to LCC, but which it was designed to successfully match. Therefore, the sentences used in Group 2 are not as similar to the sentences that exist in the knowledge base as were those of Group 1. In particular, they contain different words with the same meaning, use different tenses, use definitives and change the voice of the verbs (active vs. passive). An example sentence used in Group 2 is "Soybeans were used to make a car by Ford", while the most similar sentence in the KB would be: "Henry Ford made a car out of soybeans". In this match, the tenses are different (make vs. made), the definitive form of "make" was used in one but not the other, and, the voice of the test input is passive while that of the sentence in the KB is active. Moreover, "Ford" is qualified by "Henry". Furthermore, "soybeans" goes from being the subject in the user input to being part of the qualifying noun phrase for "car" in the KB. For those reasons, this is a more difficult match to make by the LCC algorithm. Nevertheless, albeit with greater difficulty, LCC should be able to determine that those two sentences are related and hold similar information.
3. Group 3: Sentences were purposely chosen to try to "break" the LCC matching algorithm in order to find its upper bound of effectiveness, but short of making it impossible to match. In this group, we used test input sentences that had significant differences—more so than those of Group 2—to the sentences in the KB. As in Group 2, the differences were not in the sentence structure but rather, in the length of the sentences, the voice of the sentence (active or passive) and in the words used. The test sentences in Group 3 also quantify various ways and use words instead of numbers to represent the same numeral contained in a sentence in the KB. For example, a test input is "Almost

Table 3 Accuracy as a measure of effectiveness in carrying out the correct actions in functional test, part 1-A in relation to the three difficulty groups

Group	No. of test sentences	Accuracy (%)
Group 1	30	96.7
Group 2	30	70.0
Group 3	30	43.3

half of Americans eat a sandwich every day" while what is in the KB is: "49% of US Adults eat one sandwich a day". LCC must infer that "almost half" and "49%" are the same thing. A second example test sentence is "Do you like artichoke? It is a vegetable", LCC's task was to extract the information that artichoke is a vegetable and add this to its knowledge base.

The entire list of sentences that were classified as Groups 1, 2, and 3 can be found in Appendix B of Mohammed (2019) under the Difficulty Level column. Note that some of the test sentences used as examples above were slightly changed to make the exposition here more clear.

The LCC prototype was manually given an initial knowledge base consisting of 30 sentences in the text file version of the KB. The corresponding semantic network for these sentences was built by LCC prior to the testing. These sentences were obtained from a website (Random facts, https://www.factslides.com) that contains random facts related to various topics; cars and food were used as the topics of interest in this test. The classifiers were trained prior to the start of the tests and tested as described in Sect. 4.2.3.

Note that the knowledge base was not changed between presentations of test in-puts, even though some of the earlier inputs may have involved additions and/or up-dates to the knowledge base. This was done to make the tests impervious to the order of presentation of the test inputs. However, we further note that in actual operation, additions and deletions/updates of information to the KB would happen immediately after each individual user input was processed, thereby possibly making the decisions taken later by LCC different from what resulted in these tests. The 90 test sentences and the sentences in the knowledge base can be found in Mohammed (2019)

The next sub-section describes and discusses the results of Functional Test, Part 1-A

5.1.2 Functional tests, part 1-A—results and discussion

The results are shown in Table 3. The accuracy was calculated by comparing these labels of each test statement with the actual actions taken by the LCC prototype for that statement, and dividing the total number of correct actions by 30, as shown in the formula below.

$$Accuracy = \frac{No.\ of\ correct\ answers}{30} \tag{6}$$

As expected, the accuracy of 96.7% obtained for Group 1 was the highest among the three groups. The LCC prototype only mishandled one input that was expected to be a question for the chatting statement: "can we chat?". The classifiers labeled it as information worth learning and directed the input statement to the learning system.

The accuracy for Group 2 decreased from that of Group 1, as we had expected. The LCC prototype was able to handle correctly 70% of the 30 test sentences presented. We consider this to be an acceptable result considering the increased level of difficulty of

Table 4 Accuracy measurement among the five types of actions taken by LCC

	Update/replace	Disregard	Learn	Chat	Q/A
Count	24	15	18	21	12
Accuracy (%)	82.6	73.3	72.2	75.0	83.3

this second group. A good benchmark for comparison would have been to compare these results with those of others found in the relevant literature. However, we could not find any comparable tests in the literature. Therefore, borrowing from the standard academic passing threshold of 70% as the minimum for acceptability, we used 70% as the minimum threshold of acceptable results in our work. Therefore, we consider this test of the second group to be acceptable.

The accuracy was further reduced for the third group, as it was only able to handle less than a half (43.3%) of the inputs presented. This decrease was expected, as the inputs were purposely misleading and likely to be mishandled by LCC. Most of the errors occurred when the classifiers mistakenly classified several inputs as chatting when they were not, or vice versa. For example, the sentence "I am confused and worried" in truth a chatting statement—was mistakenly classified by LCC as information worth learning.

An alternative way to assess the results gathered from this test was to combine the 90 test sentences in the three groups and report the accuracy over all the test sentences according to the correct type of action expected. Table 4 contains the accuracy measure and the number of test sentences under each action. LCC was able to handle correctly at least 72% of the text inputs in each column, which can be considered an acceptable average considering the level of difficulty of the third group.

From the observed results and considering 70% as an acceptable passing rate (as in academia), we can further infer that the main problem that faced LCC was the limitation of the classification process. Although the classification algorithm was designed to be resilient with the addition of the second level, such was not always the case, as the second level was not always activated (it only activates when there is no unanimity in the vote in the first level).

5.1.3 Functional tests, part 1-B: classifiers testing—experimental procedure

The objective of this test was to evaluate the performance of the classification process for the LCC inputs and to measure the impact of using Ensemble learning with the two levels. The test procedure involved comparing the classification results of the Ensemble learning to the performance of each of the classifiers individually.

The classifiers were all trained using the same pre-labeled dataset of 520 sentences that is the 80% training data from the 650 sentences that were discussed earlier in the classification process. LCC is designed to use only the common outputs of the three classifiers in the first level (the sentence level) when there is unanimous agreement between them on the label of the user input. When there is no unanimity among the classifiers in the first level, however, LCC activates the second level (the word level). In such cases, LCC uses the results from the majority vote of the second level only, even if they are not unanimous, and neglects the results of the first level.

Several metrics were used to report the results of Part 1-B. These are commonly used in the literature to evaluate the effectiveness of classifiers, as suggested in Provost and Fawcett (1997). In the discussion of these metrics, TP refers to True Positive statements—correctly classified as information worth learning; TN refers to True Negative statements—(correctly classified as chatting or Q/A; FP refers to False Positive—chatting or Q/A statements incorrectly classified as information worth learning; and FN referring to False Negative information worth learning statements incorrectly classified as chatting or Q/A. There was no attempt to further identify whether an input was a chatting statement or a question. Those metrics are as follows:

- *The true positive rate* (TPR): measures the proportion of the test sentences that were correctly identified as content-related information; Therefore, the higher the value, the better the classifier performance is. This is also known as the *sensitivity* or the *recall.*

$$TPR = \frac{TP}{TP + FN} \tag{7}$$

- *The false positive rate* (FPR): represents the number of negative (chatting or Q/A) statements that are wrongly categorized as positive statements (content-related). This implies that the desired value of FPR needs to be small, as that will minimize the number of test sentences that were classified incorrectly to be information worth learning while they were in truth chatting or Q/A statements.

$$FPR = \frac{FP}{TN + FP} \tag{8}$$

- *Accuracy* is the measure of the closeness of a assigned label by the classifier to the actual label.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

- *Error-rate* is the complement of the accuracy measure.

$$Error\text{-}rate = 1 - Accuracy \tag{10}$$

- *Precision*, also known as *positive predictive value*, answers what is the correct ratio of the actual positive values/label over all the positive identifications. The higher the value, the better the classifier, as it measures the number of test sentences that had been truly classified as information worth learning over all the sentences (true positive and false positive) that had been classified as information worth learning. It is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

- *F1 measure*, the harmonic mean between precision and recall

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{12}$$

5.1.4 Functional tests, part 1-B: results and discussion

A dataset of 60 input statements was used to perform this set of tests. Thirty statements contained information worth learning and 30 statements were either chatting or questions.

Table 5 Comparison of classifiers performance on dataset-1 (scaled to a range between 0 and 1)

Classifier	TPR	FPR	Recall	Precision	F1
LCC with level 1 only	0.83	0.30	0.83	0.74	0.78
LCC with level 2 activated	1.00	0.05	1.00	0.92	0.96

The statements were created by the developers and manually labeled by them as information worth learning (assigned a positive label) or not (assigned a negative label). Although the information was available, no effort was made to identify whether a negative label assignment was chatting or a question, as it was irrelevant for the purposes of these tests. Furthermore, the statements used as inputs had a unique label—that is, there were no statements with mixed labels, such as chatting and information worth learning. This was done to make the results more definitive.

The input statements were not determined a-priori to be able to trigger the second level of classification or not. Nevertheless, the results were segregated based on whether the second level became activated or not. Table 5 contains the results of the performance of the LCC Ensemble classifiers.

It is clear that the Ensemble classifier performed significantly better where the second level was activated, when compared to those cases where only the first level executed. This occurred in spite of the fact that in order to trigger the second level, the first level operations must have shown some ambiguity. Thus, it is noteworthy that the execution of the second level outperformed that of the first level, even when the classifiers in the first level were in unanimous agreement. Keep in mind that when triggered, the second level does not make any use of the results of the first level in its classification process.

5.2 Test 2: user testing

The objective of the human user tests was to determine how well the LCC prototype system would deal with real world user input and what effect this would have in what human users thought of it. This assessment was done by creating two separate groups of anonymous human test subjects. The first group of test subjects, called Group A, was asked to interact directly and mostly freely with the LCC prototype via typed text. After the period of interaction ended, they were asked to complete a survey. The second test involved asking a second, larger group of different (and also anonymous) human test subjects, called Group B, to evaluate the LCC system indirectly by reviewing the logs of the interactions of Group A subjects with the LCC system. Thus, Group B subjects did not directly interact with LCC, as did Group A subjects. Group B subjects were also subsequently asked to complete a survey to evaluate the system performance.

5.2.1 Group A test procedure

Group A consisted of 15 anonymous test subjects of age 18 or older (self-reported) who had never had any prior involvement with the LCC or with this project. The Group A participants were recruited from the undergraduate as well as graduate student population at our home university, as well as from professionals who worked outside of the university. The students were invited to interact with LCC at the Intelligent Systems Laboratory at the University of Central Florida. They were asked to use a laptop that had the LCC prototype system installed, and the actions taken by these test subjects were recorded and placed into logs. There was no time limit to the duration of the interaction period. After the interaction period ended, they were presented with a survey-type questionnaire, which they completed immediately after concluding the interactive session. Although the test subjects were met personally by us at the time of their test experience, their names or any other identifying information (other than whether they were 18 or older) was not requested, thereby ensuring anonymity. In the case of the professionals who served as test subjects, they were given the choice of either physically going to the Intelligent Systems lab on campus for their session, or to be visited by a member of our research staff who would bring them the same laptop containing the LCC prototype to their nearby place of employment. Thus, all subjects interacted with the LCC system on the same laptop computer. The Internet was not used at any time for the assessments of Group A.

Test subjects from Group A were directed to enter the system first as trustworthy users and then later as untrustworthy users. The test subjects were asked to hold a conversation with the LCC prototype via typed text that involved new information, existing information, updated information, chatting and questions. However, they were not told what to say in their conversations. More specifically, the procedure used was as follows:

The test subject was first informed about the LCC prototype system through a discussion with the lead LCC developer (the first author). Then, a brief demonstration of how to interact with the prototype was presented to the subject. This included how the prototype would add information, how to ask it a question, how to chat, and how it confirms or exchanges information. All subjects in Group A were given access to the knowledge base to reveal what information the LCC KB contained at all times throughout their interaction experience. This was done to guide their line of questioning and avoid a large number of irrelevant questions. It also served to provide the subjects with immediate feedback as to how their previous statement was handled by LCC, without the need to ask questions about it every time. The knowledge base used in the testing was composed of 30 sentences (shown in Mohammed, 2019). This number was selected to avoid overwhelming the user by having to review a large document, as well as to achieve fast response by LCC to the test input.

As mentioned above, each subject was asked to log in twice, once as a trustworthy user and then again later as an untrustworthy user. Both terms (i.e., trustworthy and untrustworthy) were explained to the test subjects before they started, and they were told what they should expect while in each mode. In the trustworthy mode, the test subjects were asked to submit statements that were new to the LCC, others that were already known, and yet others that varied somewhat from what was already in the knowledge base. This was to have them informally assess the ability of LCC to do the right thing in their opinion. In the untrustworthy mode, the subjects were asked to try to modify the knowledge base to see whether LCC permitted the modification or not. Additionally, the test subjects were asked to ask questions of the LCC and as well as to engage in a simple chat with it. The subjects were asked to perform at least 10 dialogue turns in each mode as part of the test procedure before exiting the conversation. This was important to ensure having meaningful conversations for the second group of test subjects (Group B) to review. Finally, the Group A subjects were informed that anonymous logs of their interactions with the LCC system would be saved to be used as part of the testing assessment with the Group B set of testers.

The survey given to Group A subjects consisted of 13 statements to which they were to indicate their level of agreement or disagreement. These statements were divided into four groups:

- General statements to assess what the subjects thought about conversing with LCC.

Table 6 The statements that are used in Table 7

No	Statement
1	I found the conversation with the system to be natural, as if conversing with a human
2	I found the system to be intuitive and easy to use
3	I found the user interface to be acceptable for the purpose of this program
4	I found that the system correctly added new information to its knowledge base
5	LCC correctly neglected to add information that is already in the knowledge base
6	The system correctly identified the most similar information in the knowledge base
7	The system correctly distinguish between chatting statements and content-related statements
8	I found the system was able to answer my questions
9	Transitions back and forth among the users' different inputs
10	LCC was able to correctly separate the chatting information from the related information
11	LCC prohibited me from adding and updating information in its knowledge base
12	LCC answers my questions

Table 7 Median, InterQuartile range (iqr), mean, and standard deviation for group A survey

Statement No	Median	IQR	Mean	STD
1	3	2.00	2.87	1.19
2	4	2.00	4.07	0.96
3	4	1.00	4.20	0.94
4	5	0.00	4.67	0.72
5	5	2.00	4.13	1.30
6	3	1.00	3.33	1.29
7	4	2.00	4.13	0.99
8	4	2.00	4.07	1.03
9	4	1.00	3.60	1.05
10	5	0.00	4.67	0.72
11	4	1.00	3.93	1.16
12	4	2.00	3.93	0.96

- Specific questions to assess the performance of LCC when he/she logged in as a trustworthy user.
- Specific questions to assess the performance of LCC when he/she logged in as an untrustworthy user.
- There was one open-ended question to allow the test subject to provide general feedback and any thoughts he/she might have regarding LCC that was not specifically asked in the other questions.

The subjects were asked to use a 5-point Likert scale to evaluate whether they agreed or disagreed with the statement posed in each "question", where 1 represented complete disagreement and/or total dissatisfaction with the posed statement, and 5 meant strong agreement and/or total satisfaction with the statement. The subjects could use any integer value within this range to reflect their opinions, but no fractions (e.g., 3.5).

The actual verbatim statements are described in Table 6, along with the results in Table 7.

5.2.2 Group A tests—results and discussion

We begin the discussion with the first three questions, which are general questions about interacting with the system. Question 1, which asks about the "naturalness" of the conversation resulted in relatively low user scores. A median of 3 and a mean of 2.87 indicate neutral or slightly worse user satisfaction. This is the only question of those posed whose statement can be interpreted in many different ways by the test subjects (i.e., how exactly is a conversation natural?). In reality, we do not consider the results for question #1 particularly discouraging, as conversing with a machine is still not quite the same "natural" experience that is conversing with another human. Moreover, achieving a full conversational agent was not one of our primary objectives in this research.

The test subjects' responses to questions 2 and 3 (about the intuitiveness and the user interface) were significantly better, with medians of 4 and means of slightly more than 4.0 (4.07 and 4.20). This indicates that the users found the system quite usable, an important consideration for any future commercialization of the concept.

The next six questions in the survey contain statements that seek to evaluate the opinions of the test subjects as to whether they thought that the LCC was able to make the correct decisions about how to handle the subjects' inputs in the context of a trustworthy user. The test subjects in general thought that the prototype was able to effectively add new information to the knowledge base when it was appropriate to do so (median of 5 and mean of 4.67). This was quite good. Only slightly less good was its perceived ability to discern when not to add information to the knowledge base when it was already there (median of 5 and mean of 4.13, albeit with relatively high IQR and standard deviation). Not so good was the perceived ability of the prototype to identify the most similar information in the knowledge base (question #6), with a median of 3 and a mean 3.33. This was somewhat puzzling in light of the high scores given to the statements of questions #4 and #5, knowing that to have done those first two actions well, the prototype would have also had to determine the most similar information well. The perception scores for question #7 were also good, with a median of 4 and a mean of 4.13. This belies the mixed results obtained from Test 1-b where the effectiveness of the classifiers was directly tested, as the results here are a direct consequence of classifier performance. The reader should keep in mind that unlike Test 1-b, these results are not totally objective as they assess the opinions of the test subjects. So, perhaps these inconsistencies can be attributed to the irregular and unpredictable rigor applied by the test subjects compared to the defined criteria used in the functional test. Lastly, questions #8 and #9 achieved a good score (median 4, mean 4.07 for #8 and median 4, mean 3.60 for #9) indicating its ability to answer questions to a trustworthy user as well as transition back and forth between chatting, questions and information worth learning.

With regards to the questions that pertain to an untrustworthy user, questions #10, #11 and #12 achieved good scores. Questions #10 and #12 are similar to those asked to trustworthy users (questions #7 and #8 respectively), and the results are quite good as well as comparable. This indicates that the mode of operation does not affect the users' opinion of the ability of LCC to do its job. Interesting, however, was question #11, which asks whether the system was able to keep the untrustworthy users from adding and changing information in the knowledge base, as that is an important security feature. The test subjects clearly opined that it was able to do so, with a median of 4 and a mean of nearly 4.0. Question #13—the open-ended question—provided relatively few responses and few, if any, new insights into the thinking of the test subjects beyond what is displayed in the first 12 questions. Therefore, those results are not included here.

In summary, the medians and the means for most of the 12 questions/statements in the survey were 3 or greater. The only exception was question #1, which had a mean below 3.0. Moreover, seven of the questions reported means of 4 or greater. Lastly, ten of the 12 questions had a median value of 4 or greater. These results generally reflect positive opinions of LCC system by the test subjects of Group A.

5.2.3 Group B testing procedure

We designed the group B testing to expose LCC to a larger sample of human test subjects and obtain their opinions about the LCC system and its performance. However, the Group B test procedure did not involve direct test subject interaction with the LCC prototype, but rather, they were asked to review and assess the interaction logs of Group A test subjects. This was done to facilitate the larger sample size, as these test subjects did not need to physically use the prototype deployed only on the one laptop, and participation could be widely distributed anonymously via the Internet. One hundred (100) human test subjects participated in Group B testing. All test subjects were upper division undergraduate or graduate computer science students at the University of Central Florida.

Two conversation logs from the Group A tests were presented to each Group B subject. The logs were chosen randomly from Group A participant logs and they reflected the interactions between the (two) Group A subjects and the LCC system—in effect, their conversations. The Group B subjects were also given access to the contents of the knowledge base, before and after the conversations reflected in the logs, so they could discern the differences made by the operation of LCC. However, unlike the Group A subjects, this access was not available at times between the start and end of the Group A interaction sessions.

The Group B subjects were likewise then given a survey to assess their opinions. This survey was similar to that given to Group A subjects, except that it contained only eight questions (instead of 12). The second and third questions, which related to the evaluation of the system interface were removed, as were the last two questions (11 and 12) of the Group A survey related to untrustworthy users. This was done because Group B subjects, not having interacted directly with the prototype, rendered these four questions irrelevant to them. The logs are too long to display in this paper; however, an interested reader can refer to Mohammed (2019) for a full description of these logs.

5.2.4 Group B tests—results and discussion

Similar to Group A, the results were measured by calculating the median and the Inter-Quartile Range for the 100 participants. Also computed were the mean and the standard deviation for each question. These results are shown in Table 9 and the exact statements associated with the questions of Table 9 are found in Table 8. Note that the results of the Group B tests should not be formally compared to those of Group A, as the two tests followed two very different test protocols (their experiences were very different). We next discuss each group of related questions.

Question #1, as did its counterpart in Group A results, showed a relatively low score, with a median of 2 and a mean of 2.72—below neutral. As mentioned above, naturalness was not one of our major objectives, but this is a good thing to know for future research.

The scores of the other questions all had medians of 4 and means ranging from 3.5 to slightly over 4.0 (question #4). These are good results in our opinion. In spite of the fact that we stated previously that a direct comparison between these results and those of

Table 8 The statements that are used in Table 9

No	Statement
1	I found the conversation with the system to be natural, as if conversing with a human
2	I found that the system correctly added new information to its knowledge base
3	LCC correctly neglected to add information that is already in the knowledge base
4	The system correctly identified the most similar information in the knowledge base
5	The system correctly distinguish between chatting statements and content-related statements
6	LCC was able to answer questions
7	Transitions back and forth among the users' different inputs
8	The system was able to correctly separate the chatting information from the related information

Table 9 Median, Interquartile range (IQR), mean, and standard deviation (STD) for group B (100 participant) survey

Statement No	Median	IQR	Mean	STD
1	2	2.00	2.72	1.44
2	4	2.00	3.78	1.08
3	4	2.00	3.93	1.18
4	4	1.00	4.16	1.00
5	4	2.00	3.55	1.19
6	4	2.00	3.69	1.14
7	4	2.00	3.61	1.17
8	4	2.00	3.56	1.19

Group A should not be made, it is unavoidable to notice that the Group B results generally track the Group A results fairly well for the same questions, although somewhat lower. This serves as an informal confirmation of both sets of results.

Overall, the results across Group B indicate satisfaction with the LCC on the part of the test subjects, as the median is 4 for all the questions except the first one, which had a median of 2.

5.3 Test 3: complexity and scalability assessment

To assess the runtime complexity of the LCC algorithm, we performed an empirical analysis by measuring wall clock runtime over various knowledge base sizes. More specifically, we measured how long it takes LCC to respond to the user inputs for varying sizes of the knowledge base. The time taken to produce a response by the system is thought to be proportional to the size of the knowledge base because the larger the knowledge base, the more search that is required to relate the user input to the most similar information in the knowledge base. Our question was whether this relationship would be linear or non-linear, and if linear, how steep would the curve be.

The scalability was measured with the knowledge base size ranging from 30 sentences to 960 sentences. Those sentences were specifically selected to be closely related and have many connections in the semantic network so as to make the conditions more demanding in terms of search.

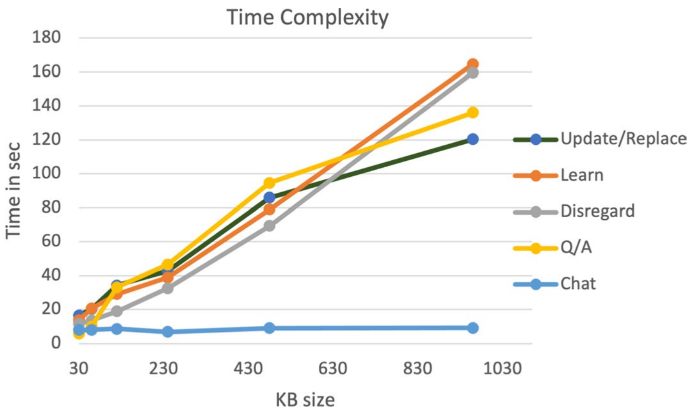


Fig. 9 Scalability of the LCC system

We measured the time it took for each of the operations to produce a response: add/learn, update/replace, confirm/disregard, question-answering or chatting. The results were averaged over five runs for each KB size. The actual sentences used can be found in Mohammed (2019). Figure 9 shows the means of the various operations (adding, exchanging, confirming, question-answering and chatting) averaged over five runs. The x-axis represents the number of sentences in the knowledge base, while the y-axis represents the average time it took LCC to respond to the user input in seconds of wall clock time.

From the results of Fig. 9, it is clear that the system spent a nearly constant amount of time responding to the chatting statements across all the sizes of the knowledge base. This was fully expected, as the chatting engine does not use the knowledge base and thus, its time to respond to a chatting input has no relation to the size of the knowledge base.

From Fig. 9 we can see that LCC takes longer to respond to a Q/A request, a confirm operation, an exchange operation and an add operation, as the knowledge base increases in size. As the figure shows, the proportion of time vs. KB size is definitely linear for all these functions. At the upper end of the KB size, the time responses were in the order of two+ minutes, which is troubling in absolute terms, as no user would willingly put up with such lengthy delays. This points to an area where significant improvement will be necessary in future research. However, for small and medium sized KBs, the performance is deemed acceptable, as there were no indications in Question #13 of intolerance on the part of the test subjects in the Group A user-based testing described above that used small knowledge bases.

As a note to the interested reader, a theoretical derivation of the algorithmic complexity of the LCC overall algorithm can be seen in Mohammed (2019) It also resulted in a linear complexity.

6 Summary and future work

This research explored the idea of how a computer agent can learn new information from a “natural” conversation with a human, similar to how humans learn from each other through extended dialogues. In this paper, we present a system called Learning from a Casual Conversation (LCC) that involves direct human interaction with a learning agent through a natural language dialogue to allow the agent to acquire new information. This research can have an impact on improving the breadth, depth and accuracy of a computer agent's knowledge base, by allowing the agent to learn new information without the need to involve a programmer to update its knowledge base.

LCC operates by combining multiple components to form the overall architecture of the system. The most important and fundamental components of LCC are the classifier unit, the learning unit, and the knowledge base unit. The classifier unit is responsible for labeling the user input as either a chatting statement, a question or as a statement that contains information worth learning. The learning unit determines exactly what, if anything, is to be learned from the human input passed to it by the classifier. The knowledge base unit holds the information "known" by the agent in the form of a semantic network linking together the sentences that make up the agent's information.

An LCC prototype system was built, and assessed using various tests. The overall results showed that it was effective in learning what was to be learned. It also showed good user acceptance of the concept and of the prototype itself. This all indicates that this approach is a significant step towards extending a computer agent's ability to learn through a novel means.

There are many possible applications for LCC. For example, it can be used for social companions where the human user can share information with a robot or an avatar using natural language. LCC can also be applied to help human-robot assistants such as Alexa and Siri to automatically update their knowledge base without the involvement of a developer. Finally, we should note that we do not claim that our results are generalizable, but only apply to the specific prototype we developed, the tests we executed, the training data we used and other such very specific decisions taken in our design

We found several areas where future research is warranted. One major area of improvement is having the LCC "understand" the meaning of the user input statements. The current version is limited to using similarity measures to determine how similar/different the words/sentences in the user's input are from what already exists in the KB, also in the form of sentences. Therefore, application of natural language understanding could be used to improve the learning process significantly.

Another possible enhancement would be to use voice instead of text to communicate with the system. This would allow for an easier and a more natural interaction with the system. Moreover, it can help users with disabilities who cannot use a keyboard to communicate with the agent. Such an enhancement would require additional work to deal with errors in the speech recognition system as well as adding text-to-speech capability for responding to the user in speech.

Trust evaluation of the users and of the credibility of the human source of information would be an important enhancement to LCC, mainly for when the system could potentially be used by less-than-competent and/or by malicious users to unwittingly or purposely corrupt its knowledge base.

Improving the classification process also represents a major need for improvement, as the current approach leads to relatively large number of misclassifications. We also hope to explore the use of online training for the classifiers (the current training was performed offline). Continually on-going training that would take place as the system operates would allow the classifiers to learn the new patterns that a particular user employs, which could improve the classification process in general.

Improving/enhancing the "naturalness" of the conversation will also be undertaken in future research, as that was the one statement that stood out as being deficient in the human test subject scores. Lastly, the 12 simplifying assumptions stated in Sect. 2 must be lifted in order to make a future LCC operational in the real world.

Author contributions AMA: Conceptualization, methodology; investigation; software development: data acquisition and curation; writing and editing. AJG: conceptualization; project administration; supervision: manuscript editing

Funding Work was supported indirectly through a teaching assistantship from the University of Central Florida for the first author.

Availability of data and materials All used data can be found in the appendices of the first author full dissertation document that can be accessed from https://stars.library.ucf.edu

Code availability Not applicable at the moment but the authors are planning to publish the code soon

Declarations

Conflict of interest There are no conflicts of interest for any of the authors

Ethical approval The use of human test subjects and surveys were approved by the Institutional Review Board at the University of Central Florida, SBE-18-1418 dated: 8/14/2018. The approval letter can be found in APPENDIX H from the frist author dissertation document that can be accessed from https://stars.library.ucf.edu/etd/6297/. The authors consent that the submitted work is original and have not have been published or submitted elsewhere.

Consent to participate Not applicable as the authors did not use any identification data related to the participants in this research.

Consent for publication Not applicable.

References

Anderson, J. R. (1996). ACT: A simple theory of complex cognition. *American Psychologist, 51*, 355.

Apté, C., Damerau, F., & Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *Assoc Comput Mach (ACM) Trans Inf Syst TOIS, 12*, 233–251.

Chacón, A., Marco-Sola, S., Espinosa, A., Ribeca, P., & Moure, J. C. (2014). Thread-cooperative, bit-parallel computation of levenshtein distance on GPU. In *Proceedings of the 28th of international conference on supercomputing* (pp. 103–112)

Chang, Y.-W., Hsieh, C.-J., Chang, K.-W., Ringgaard, M., & Lin, C.-J. (2010). Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research, 11*, 1471.

ChatterBot-machine learning, conversational dialog engine. Retrieved from, https://chatterbot.readthedocs.io/en/stable/. (2019).

Chieu, H. L., & Ng, H. T. (2002). A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of the association for the advancement of artificial intelligence (AAAI),* (vol. 2002, pp. 786–791).

Clark, H., & Schaefer, E. (1989). Contributing to discourse'cognitive. *Science, 13*(13), 259–294.

Cox, G. (2017). chatterbot.corpus.english.greetings. Retrieved from, https://github.com/gunthercox/chatterbot-corpus/blob/master/chatterbot_corpus/data/english/greetings.yml.

Cox, G. (2019). chatterbot.corpus.english.conversations. Retrieved from, https://github.com/gunthercox/chatterbot-corpus/blob/master/chatterbot_corpus/data/english/greetings.yml.

Dai, W., Xue, G.-R., Yang, Q., & Yu, Y. (2007). Transferring Naïve Bayes classifiers for text classification. In *Proceedings of the association for the advancement of artificial intelligence (AAAI)* (pp. 540–545).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805

Dietterich, T. G. (2002). Ensemble learning. *The handbook of brain theory and neural networks* (vol. 2, pp. 110–125).

Dunford, R., Su, Q., & Tamang, E. (2014). The Pareto principle. *The Plymouth Student Scientist*, *7*, 140–148.

Eggins, S., & Slade, D. (2004). *Analysing casual conversation*. Equinox Publishing Ltd. Cassell.

Feldman, A. (1959). *Mannerisms of speech and gestures in everyday life*. New York, NY: International Universities Press.

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 363–370).

fuzzywuzzy. Retrieved from, https://github.com/seatgeek/fuzzywuzzy

Ganesan, K., Zhai, C., & Han, J. (2010). Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 340–348).

Garfinkel, H. (1967). *Studies in ethnomethodology*. Prentice Hall.

Gilmartin, E., Saam, C., Vogel, C., Campbell, N., & Wade, V. (2018). Just talking-modelling casual conversation. In *Proceedings of the 19th annual SIGdial meeting on discourse and dialogue* (pp. 51–59).

Goldberg, Y., & Levy, O. (2014). word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv:1402.3722

Goldwasser, D., & Roth, D. (2011). Learning from natural instructions. In *Proceedings of international joint conference on artificial intelligence (IJCAI).*

International workshop on semantic evaluation (SemEval-2015). http://alt.qcri.org/semeval2015

Jaccard similarity measure. Retrieved from, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_score.html

Joseph, V. R., & Vakayil, A. (2021). SPlit: An optimal method for data splitting. *Technometrics, 64*, 166.

Kuhlmann, G., Stone, P., Mooney, R., & Shavlik, J. (2004). Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *Proceedings of the association for the advancement of artificial intelligence (AAAI) workshop on supervisory control of learning and adaptive systems*.

Li, J., Miller, A. H., Chopra, S., Ranzato, M., & Weston, J. (2016). Learning through dialogue interactions. arXiv:1612.04936

Liu, B., & Mazumder, S. (2021) Lifelong and continual learning dialogue systems: Learning during conversation. In *Proceedings of AAAI*.

Luong, M.-T., Pham, H., & Manning, C. D. (2015) Effective approaches to attention-based neural machine translation. arXiv:1508.04025

Mihalcea, R., Corley, C., & Strapparava, C. (2006) Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the association for the advancement of artificial intelligence (AAAI)* (pp. 775–780).

Mohammed, A. A. (2019). Machine learning from casual conversation. Doctoral Dissertation, Department of Computer Science, University of Central Florida Electronic Theses and Dissertations. 6297. Retrieved from, https://stars.library.ucf.edu/etd/6297

Naïve Bayes text classification. Retrieved from, https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. In *Proceedings of international joint conference on artificial intelligence IJCAI-99 workshop on machine learning for information filtering* (pp. 61–67).

Provost, F. J., & Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. Knowledge Discovery and Data Mining (KDD) (pp. 43–48).

Random facts. Retrieved from, https://www.factslides.com

Rybski, P. E., Yoon, K., Stolarz, J., & Veloso, M. M. (2007). Interactive robot task training through dialog and demonstration. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction* (pp. 49–56).

Sacks, H., Schegloff, E. A., & Jefferson, G. (1978). *Studies in the organization of conversational interaction* (pp. 696–735). Elsevier.

Sultan, M. A., Bethard, S., & Sumner, T. (2015). DLS @ CU: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th international workshop on semantic evaluation* (SemEval 2015) (pp. 148–153).

Sultan, M. A., Bethard, S., & Sumner, T. (2014). Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics, 2*, 219–230.

Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral cloning from observation. arXiv:1805.01954

Torrey, L., Walker, T., Shavlik, J., & Maclin, R. (2005). Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *The European conference on machine learning and principles and practice of knowledge discovery in databases (ECML-PKDD)* (pp. 412–424).

Traum, D. R., & Hinkelman, E. A. (1992). Conversation acts in task-oriented spoken dialogue. *Computational Intelligence, 8*, 575–599.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Ventola, E. (1979). The structure of casual conversation in English. *Journal of Pragmatics, 3*, 267–298.

Weston, J. E. (2016). Dialog-based language learning. In *Advances in neural information processing systems* (pp. 829–837).

WordNet. Retrieved from, https://wordnet.princeton.edu/

Yujian, L., & Bo, L. (2007). A normalized Levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*, 1091–1095.

Zhang, H., Yu, H., & Xu, W. (2017). Listen, interact and talk: Learning to speak via interaction. In *NIPS workshop on visually-grounded interaction and language*.