Systems Engineering and Conversational Agents

James O'Shea, Zuhair Bandar and Keeley Crockett

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Chester St., Manchester M15GD, United Kingdom
{j.d.oshea, z.bandar, k.crockett}@mmu.ac.uk

Abstract. This chapter describes Conversational Agents (CAs) in the context of Systems Engineering. A CA is a computer program which interacts with a user through natural language dialogue and provides some form of service. CA technology has two points of interest to systems engineers: the use of systems engineering techniques in CA research and the application of CAs in projects development. CAs offer the opportunity to automate more complex applications than are feasible with conventional web interfaces. Currently such applications require a human expert in the domain to mediate between the user and the application. The CA effectively replaces the human expert. This chapter reviews the current capabilities of various CA technologies, outlines a development methodology for systems engineering practitioners interested in developing real world applications and suggests a number of directions for systems engineers who wish to participate in CA research.

Keywords: Conversational agent, systems engineering, dialogue, evaluation, methodology, semantic similarity, short text

Section 1 Introduction

In the early decades of computing ordinary people did not have any interaction with computers at all, any information that was needed for an application was entered by specialised clerks. This was followed by a period (largely driven by the internet) in which users could drive simple applications by entering simple facts and figures directly. Recently an explosion of internet use has demanded that ordinary people interact with complex applications, and the trend is for these to be of increasing complexity. Such applications would normally require a domain expert to interview the user to obtain the necessary information.

Automating such processes represents a serious challenge to computing practitioners. This chapter describes a possible way forward, Conversational Agents (CAs). CA technology has two points of interest to systems engineers: the use of systems engineering techniques in CA research and the application of CAs in projects development.

A CA is a computer program which interacts with a user through natural language dialogue and provides some form of service.

Typically this dialogue system serves a business goal such as providing information, advice or selling. A suitably-designed CA plays the role of the human expert and is generally in charge of the conversation; such CAs are fundamentally intelligence-based systems.

The idea of a computer taking the role of a human in conversation was first proposed by Alan Turing [1], as a test of machine thought. Although there has been substantial philosophical debate about the Turing Test [2-4], it has no impact on the validity of CAs. Practical CAs aspire to provide the user with the kind of advice or services that would come from a knowledgeable or experienced human, but in a purely behavioural sense. This form of CA presents with "Intentionality", that is it displays beliefs, desires and intentions concerning objects, events and states of affairs in the real world [5] – but it is not required to have a "mind."

The applications that have been proposed for practical CAs include health care dialogue systems [6], real estate sales [7], phone call routing [8], intelligent tutoring [9] and natural language interfaces to databases [10, 11].

Probably the most effort has been expended on CAs for online customer self-service, which provide the user with the kind of services that would come from a knowledgeable or experienced human. In 2005 there were at least 10 major companies operating in this area, including IBM and strategic partners of Microsoft [12]. At least 28 patents have been registered concerning CAs and closely related technologies. Despite this effort, success has been mixed and more research will be required to achieve the goal of a functional CA which can fill the role of a human [6].

Section 2 The scope of CAs

The term "CA" can have a very broad scope including:

- Spoken Dialogue Systems
- Chatterbots
- NLP-based Dialogue Management Systems
- Goal-Oriented CAs
- Embodied CAs

2.1 Spoken Dialogue Systems

Spoken Dialogue Systems (SDSs) are concerned with the conversion of speech into text. The average user might expect to interact with a CA by speaking to it directly and having the speech interpreted by SDS algorithms. In fact the field is insufficiently developed for this to be practical for anything but trivial applications. This is due to the relatively high error rates involved in converting the audio input into text. The performance of SDSs is usually measured as the Word Error Rate (WER) which takes account of the numbers of insertions, deletions and substitutions needed to correct a transcribed segment of speech [13].

Consequently, work on SDS systems falls into two categories.

The first covers systems which can convert speech from members of the general population and the second covers systems which are trained to recognise speech from a particular speaker.
Systems which cover the population split into two further categories, small vocabulary and large vocabulary.
Small vocabulary systems, in use since the 1990s for applications like paying bills, need to recognise the digits plus a few other words such as “account”. These systems are capable of recognising tens of words and can achieve WERs of less than 1% under ideal conditions [14].
Large vocabulary systems contain tens of thousands of words [15] and represent the ideal interface for a CA. Unfortunately such systems have high word error rates, examples being a range of 18.4% -35.5% [16] and a particular WER of 25% [17].
Although small vocabulary systems are in routine use the individual words are simply matched as symbols and no real conversation takes place. More research is needed to improve the WER of larger vocabulary systems to make SDSs truly useful for CAs.

2.2 Chatterbots

Chatterbots are the direct outcome of attempts to create a system that would pass the Turing Test and are also stimulated by the Loebner prize which offers a substantial cash prize for passing a version of the test. The objective is to pass as a human for a limited period of time. Consequently chatterbots are programs that engage a human in social conversation and attempt to prolong the conversation for as long as possible.
Chatterbot development is driven by a “cat and mouse” game between developers and judges. The chatterbot is considered successful if it can prolong a conversation, no matter how banal or purposeless, for the time period without being detected as a machine by a judge.
The dominant technology in chatterbots is Pattern Matching. This approach requires scripts that define the conversation to be executed by a pattern-matching engine. The scripts contain rules which in turn contain patterns. The chatterbot responds to a user utterance based on the best match to one of its patterns.
Chatterbot developers program tricks into their systems to convince the user (as a substitute for real thought) and when users are in a judging mode they indulge in unnatural antisocial behaviour to “out” the chatterbot [18].
Although the technology behind chatterbots may be useful as a component of CAs, the chatterbot in itself is too limited to have use as a practical CA.

2.3 Natural Language Processing based dialogue management systems

Natural Language Processing (NLP) is largely concerned with document retrieval, information extraction, and text categorisation [19].
There is an established interest [20] in applying established NLP procedures such as using parsing, keyword extraction and formation of a structured lexicon to systems which engage in dialogue.

In initial work there was a lack of substance when it came to reasoning about the meaning of user utterances and the production of relevant responses. Modularisation (or compartmentalisation) of NLP based systems leads to these problems being lumped together as Natural Language Understanding (NLU).

The dominant approach to NLU is the frame-based system [21-23]. This is effective for simple applications such as making bookings for journeys or theatre seats. A related approach is the use of state-based systems, popular in healthcare [6]. These undergo state transitions triggered by the content of user utterances. Some success has been achieved with limited systems in which tight constraints are placed on the utterances that the users can produce. This can be done with forced choice questions (e.g. yes or no answers) or the detection of a very restricted set of highly salient speech fragments; however the dialogue may be unnatural. More flexible dialogue is possible, using more powerful grammars and probabilistic/empirical techniques, but is not trusted when high accuracy of understanding of the user intent is required [6].

The most promising NLP-based approach (used within a CA) currently being investigated, at the University of Cambridge, uses phrasal grammar rules to extract the dialogue act type and a list of attribute/value pairs from each utterance and a template-based response generator [24, 25]. However, this approach has only been evaluated in the laboratory, with a simple domain, Towninfo, which recommends restaurants, museums and similar destinations of interest to tourists.

Despite the considerable effort put into NLP, it has a number of problems for use in real-time CAs. The first is whether the chains of computationally intensive processes involved will scale up to real-word applications deployed on the web, especially when large numbers of users are involved. Secondly, each process has a particular error rate and the cumulative effect of these may affect the classification of user utterances. Thirdly, NLP relies on grammatically correct sentences, yet most user utterances are not properly-formed. Repair processes to remedy this incur a further computational overhead. Fourthly, research into NLP is fragmentary in nature. For example recent work has focussed on monitoring the human's engagement [26, 27], interaction control [28, 29] or determining if a party is being addressed [30]. What is really required is a concerted effort to produce a less sophisticated, but functional, system.

2.4 Goal-Oriented CAs

A Goal-Oriented CA has a deep strategic purpose in holding the conversation and its design incorporates mechanisms that enable it to focus the conversation on achieving a goal. This is what distinguishes it from a Chatterbot.

The original design objective of chatterbots was to prolong social chit-chat, thus they are easily de-railed by human users when used for practical applications.

A Goal-Oriented CA (GO-CA), on the other hand, is specifically designed to interact with a human, using natural language dialogue, to achieve a particular business goal - such as providing information, advice or selling. It plays the role of an empowered human in a productive application or task. Thus the GO-CA [31] may spend more time leading the conversation and asking questions than the human.

In general terms the human approaches the GO-CA with a problem or need. In current implementations [31] a pattern matching dialogue front end is combined with a rule-based system, which contains a model of the problem domain that is expressed in terms of a set of attributes. Through the process of dialogue, appropriate attributes are captured to model the particular problem experienced by the user and identify the appropriate solution.

The GO-CA is a mixed-initiative system (from time to time either the human or the agent may take control of the conversation). Due to the goal-oriented nature of the agent it will take the initiative in the first instance and will always return to the goal after the human has diverted the conversation (for example to ask for a clarification of something said by the agent).

Figure 1 shows the generic architecture for a typical GO-CA [31]. This is intended to take on challenging real-world applications in which the human user may present adversarial, disruptive or deceptive behaviour at times during the conversation.

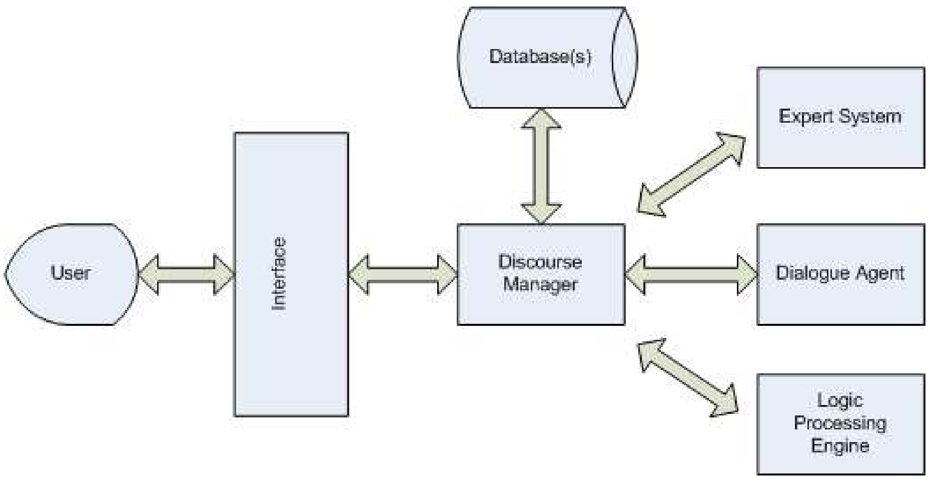

Fig. 1. Goal-Oriented CA architecture

Modularisation is an important element of the architecture. New modules can be added for extended functionality or existing ones omitted if not required. In this particular CA the rule-based system takes the form of an expert system. In another the application required an interface that allowed the agent and the users to communicate by SMS text messaging.

The architecture is best described by considering a dialogue transaction between the agent and a user.

1. The first step in a dialogue transaction is for the expert system to identify the attribute (or next attribute) whose value is to be captured.
2. The expert system passes this requirement to the Discourse Manager (DM).
3. The DM invokes the dialogue agent (DA) which produces a suitable utterance for the agent and returns this to the DM.
4. The DM passes the agent’s utterance to the web interface which sends it to the user.

5. The user replies to the agent and the web interface passes the user utterance to the DM.
6. The DM invokes the DA to determine if it contains the required attribute.
7. If the attribute has not been captured, go to step 3.
8. When the attribute has been captured, the discourse agent passes the attribute to the expert system, which updates its model.
9. If further attributes are required by the expert system, go to step 1.
10. The solution to the problem is communicated to the user.
11. At this stage the user may end the conversation or continue by asking clarification questions.

The interface can consist of a text-based, instant messaging-style system which is very familiar to users of social networking applications. It is also possible to use a speech recognition system (currently this would need to be trained for a specific user; large vocabulary word recognition may be feasible in the future).

The database is used for long-term storage of user attributes (e.g. start date with employer) and the logic processing engine provides domain-specific computational tasks (e.g. date calculations).

The common feature between GO-CAs and chatterbots is the prevalence of the technique of pattern matching. However a GO-CA will engage in extended dialogue, during the course of which it will appear to have mental states that are directed at states of affairs in the world e.g. beliefs, desires, hopes, perception, intention etc. Whereas chatterbot-based systems typically present a business’s FAQ list with a human face, GO-CAs are intended to give sophisticated advice on topics such as bullying and harassment in the workplace [31].

2.5 Embodied CAs

An Embodied CA (ECA) is characterised by a multimodal interface which includes a facial display, hand gestures, posture etc., interaction with a human (or representation of a human in a computer environment) and a dialogue system where both verbal and nonverbal devices advance and regulate the dialogue between the user and the computer [32].

The degree of embodiment can vary considerably. At its simplest it involves a graphic representation of the agent capable of facial expressions, where the intention is to provide a generally heightened sense of realism. One example of this approach is a virtual museum guide used to investigate the kind of dialogue that embodied agents provoke from humans [33]. The most advanced view of embodiment encompasses facial expressions and gestures by the agent coupled with the reading of gestures from the user [32, 34, 35]. This extends to the modelling of emotions on the part of the agent [35].

Whilst there is clear potential for embodiment to improve GO-CAs, for example through disambiguating pronouns such as *this* and *that* using pointing gestures and shared visual space [32, 36], the dialogue tasks attempted remain relatively simple.

The REA agent [37] uses chatterbot-style social engagement combined with a linear attempt to collect a very small number of attributes in order to make a property recommendation. ECAs are generally used with very low-stakes applications, where

there is little to lose if the agent fails to operate correctly, for example museum guides [33, 34, 38].
Research continues in this field, but tends to follow fragmentary and specialised interests rather than focussing on the holistic problem of building a GO-CA for real-world applications [27, 29] [39]

Section 3 Practical Applications of CAs

The primary interest of this chapter is the development of CAs for real-world applications. There are two requirements that a CA technology must meet to achieve this:

- It must be capable of handling extended dialogue about complex tasks
- It must be capable of operating in real-time when deployed over the internet.

We begin this section with an overview of the use of the various forms of CA to selling. This is followed by a description of a more challenging application where the stakes are higher, leading to more complex user behaviour.

3.1 CAs for selling

Selling is an obviously useful activity and customers often have difficulty in making choices about a purchase, requiring assistance from a human sales person.
The move to selling over the internet reduces costs but removes this human element. A CA can rectify this problem. Selling is a challenge that can be met at a number of different levels of complexity, which makes it a useful vehicle for comparing CAs. Also, in its more challenging forms it creates work for practitioner systems engineers which has a clear value for their business clients.
From the systems engineering research point of view it is sufficiently demanding as a test of new theories, processes and tools, and it has been attempted across the scope of CAs described above, from SDSs to embodied CAs.
Influential early work in selling using SDS was stimulated by the DARPA air travel challenge [40]. Attributes collected from the user were origin, destination, date (of flight), time (of flight), airline, trip-type (one-way or return), hotel and car. The attributes were collected in a linear fashion and there was no interaction between them, apart from asking whether ground arrangements were required before asking about the hotel and car requirements [41]. Walker also used a restaurant booking application to investigate multi-attribute decision making. In this case the restaurants were modelled using 6 attributes: *food quality, cost, decor, service, food type* and *neighbourhood.* The attributes were scored on a scale and the combination of scores was used to make recommendations [42].
Recent work on SDS introduces more sophisticated processing of the user utterances, for example with more support of user barge-ins (interruptions) [43]. But current work confirms the limited number of attributes that can be managed using an SDS system [44].

The current position is that the SDS approach is limited to transactions that require a small number of factual answers and these are characterised by situations where the user has already made the decision to buy, for example in booking travel arrangements. Real-world systems have been deployed effectively for the payment of utility bills where only recognition of digits and a few additional words is required to make a payment.

SDSs currently fail to meet the requirement to handle more extended dialogue, with the general population, principally because of the high word error rate with large vocabularies. Current SDS technology does not achieve a suitable word error rate for speech from the general population for the size of vocabulary needed for more sophisticated applications.

A significant number of chatterbot systems have been developed for selling over the past couple of decades. Furthermore, the chatterbot is the only form of agent to have been deployed on large companies’ websites for real interaction with the general public. Examples include Hank (Coca Cola), Kate (Ford) and Anna (IKEA) [45]. Of these agents Hank and Kate have failed to prosper. They have been replaced on the websites by FAQ systems with question entry boxes and these do not encourage the user to enter into dialogue. The IKEA bot, Anna, still has a visible presence on the IKEA website. Anna presents as a 2-D cartoon head and shoulders figure. She is a sales assistant who accepts typed input and replies with text, which is also spoken by a female voice synthesiser. The agent blinks, smiles and makes occasional posture changes but can not be considered as truly embodied as these gestures have no significance in the conversation. In terms of dialogue Anna is extremely limited, simply pointing customers to the IKEA catalogue, for example:

```
User: I'm having a barbecue

Anna: I'm really not sure what it is you're trying to
say. Can you please try and re-phrase your question or
statement.

User: do you sell barbecue accessories

Anna: EHere you will find the Decoration Accessories
Category.

User: do you sell plastic glasses

Anna: Please have a look at the Glasses Subcategory.
```

Under these circumstances a human sales agent could have told an anecdote about organising a barbecue, gone through specific examples of the utensils required and prompted the purchase of additional items, so clearly an opportunity has been missed.

Selling chatterbots are still under development, a recent example being Susan, hosted on the Kegel Motorcycles website [46]. Susan is described in the chatbots.org site as “. . . an attractive, smart and bright cowgirl who not only talks with clients but also presents multimedia.” Susan appears on the website as more conservatively dressed

and a little more sober in manner than the original description. Susan is composed from video clips of a real woman speaking the dialogue. Susan uses a few gestures (pointing to self, pointing to menus, dialogue box etc.) and expresses boredom if there are pauses in the conversation (for example by tapping on the glass of the monitor as if it were a window). Despite appearances, the behaviour is quite superficial and is a long way from qualifying as genuine embodiment.
Furthermore, the interaction hardly qualifies as dialogue because it is so constrained. The opportunity to exploit social chat in the selling has been sacrificed to achieve robustness. Susan repeatedly refers the user to a menu below the dialogue box, there is an overall menu of topics covered by the side of the agent and during typing an auto-correct style “did you mean . . .” steers the user towards one of Susan’s standard replies. For example:

```
User: I’m 55, what sort of bike should I get?

Susan Try the menu buttons on the left
```

The menu displayed to accompany this includes much off-topic information such as company history, financing deals etc.
Real-world applications require the capture and analysis of attributes from the user’s utterances. It should be noted that neither Anna nor Susan do this to any noticeable degree and are thus less analytical than the SDS DARPA travel systems.
The one strength of the chatterbot is social chat. The original instantiation of Susan seemed to encourage this as the video clips were of a pretty, flirtatious cowgirl. Whilst this is probably consistent with the brand image required for a motorcycle vendor, it also underlines the weakness of chatterbot technology. Experience in developing CAs shows that users are likely to be attracted to this kind of site for two reasons. The first is to flirt with the chatterbot. The second, particularly with such a human representation, is to test it to breaking point (much like the behaviour of Loebner prize judges).
Social chat creates a paradox at the heart of chatterbot systems. On one hand the social chat capabilities improve the user’s experience, which ought to make chatterbots more effective as sales agents. On the other hand purposeless chat does not progress the conversation to achieve the client’s objectives.
Conventional chatterbots are not equipped with the necessary mechanisms to achieve both requirements, hence the development of the GO-CA.
Selling has been of interest to the NLP based community for the past two decades as exemplified by the SCHISMA theatre bookings project [47]. Schisma projects began with text-based interfaces and an intention to move to SDS interfaces “in the near future” [48]. Progress may be judged by recent work which combines NLP techniques with an SDS interface. The NLP element is sophisticated, involving semantic parsing, context free grammar and dialogue act recognition. The SDS component however, is only capable of recognising 263 words (trained using data from British speakers).
The theatre booking task is a little more sophisticated than the SDS systems described earlier. Attributes to be captured include actors, authors, performances (category theatre/opera as well as title), dates and venues. This entails the recognition of data

types such as number, date, time etc. in the utterance, which is performed with error correction before the parsing stage.

Although this work is interesting, it is only partially implemented and the research still makes use of the Wizard of Oz approach (in which a human simulates the CA) for collecting corpus data.

Of all the techniques, one might have expected NLP to produce working, robust, systems - however these do not exist in the real world. There are two principal reasons for the failure of NLP to produce. First, NLP requires chains of processes such as stemming, pos-tagging, syntactical repair and parsing which can lead to cumulative errors in recognising a user utterance. Disambiguation is a further problem as the usages and senses of English words are not easily identified. To make things worse, as observed by Donald Michie, "Real chat utterances are mostly unparsable. They are concerned with associative exchange of mental images. They respond to contextual relevance rather than to logical or linguistic links."[49].

The second problem is the high computational complexity of NLP processes which raises serious questions about its scalability for a CA deployed over the web serving multiple users. For example the UK national flu service received 9.3 million hits per hour on the first day of operation (resulting in it crashing even though this was a simple menu-based system)[50].

GO-CAs have also been developed for selling; one such is VSA [51]. The system is described as dialectical and goals are driven through an internal process of argument about logical formalisations of the dialogue. The sales process is divided into 3 phases, before sale (identifying needs and suitable products), sale (negotiation to make a deal) and after sale (which was not dealt with in Morge's study). The scope of the system is not clear, but one example dialogue in which a quilt is sold has 4 attributes, allows the user to barge-in, volunteering information and also allows the user to question the agent. Although the agent was developed using commercial software, there is no indication of any real-world deployment.

Although the chatterbots like Anna and Susan present a human face, these are not genuinely embodied. The distinguishing feature of embodied CAs is that the gestures and expressions are purposeful in contributing channels of information to the dialogue. An early and well-known ECA is REA, the real estate sales agent [32]. REA is indicative of the greater interest in embodiment than dialogue in ECA research. A typical study using REA captured 3 pieces of information in a linear sequence: the city the user wanted to live in, the number of bedrooms desired and acceptable level of rent. The objective of the study was to investigate whether the use of social language fostered trust in the agent on the part of the user. Current work continues to take a strong interest in social behaviour, such as the display of emotions [52], rather than deployable systems.

ECAs are still relatively immature. They build embodiment on top of existing dialogue management techniques and inherit their weaknesses. They have not been used to tackle the kinds of applications of interest in this chapter and consequently their potential advantage (disambiguation through gesture) has not been put to a serious test. The leaves the GO-CA is the most currently promising technology for developing real-world applications. The following section describes a GO-CA which ran for 8 years, advising university students about debt problems.

3.2 A GO-CA student debt advisor

Adam, the student debt advisor, operated from 2002 to 2010. Adam was designed for a very specific application, to assist a student who had received a warning letter about debt to the university. An important distinction between Adam and a chatterbot system is that Adam was not intended to counsel students about their feelings about debt, rather to follow the steps necessary to pay the debt off.

Adam's rule-based system uses 23 different debt-based attributes, as well as collecting some information not used for decision making, such as the student name.

By analysing combinations of the attributes, Adam directs the student to one of 26 different actions to solve the debt problem. Analysis is performed by a decision tree, so only the subset attributes required to traverse from the root to the appropriate leaf need to be collected. There are many different routes through the tree with corresponding dialogue structures.

One outcome of the knowledge engineering phase is that the tree is designed to process and complete the most frequent solutions first, reducing the computational load on servers in the deployed system.

For example, in the previous telephone-based system, one of the most common calls was from students who had paid off the debt and wanted assurance that no further proceedings would be taken. This situation is caught with the attribute Have_paid_already in the first conversational context.

The many conversational contexts within the Dialogue Manager are decomposed into 4 groups:

- Conventional dialogue
- Filter
- Oracular Layer
- Aliza Layer

The contexts making up the Conventional Dialogue put the questions to the user to acquire attributes, perform clarification tasks and provide the instructions (diagnosis).

The filter is executed every time the user types an utterance, regardless of the current context in the main dialogue. It performs two tasks. First, it contains a small number of rules to detect highly obvious statements of the values of any of the 23 attributes. So if a user volunteers additional attributes in the current context, they will still be recognised and captured. Secondly it is used to detect racist or other highly offensive language in the conversation (which results in the conversation being terminated).

The oracular layer is responsible for answering questions put to the system. These include the many possible requests for information or clarification required to supply the attribute values to Adam, as well as general questions.

The Aliza layer (named for its resemblance to the Eliza chatterbot) layer contains general chat. This allows Adam to respond to social remarks included in the conversation. It also includes light-hearted responses to personal, challenging or antisocial remarks by the user.

A user utterance is first passed through the Filter and then (if the desired attribute is not captured) to the Conventional Dialogue layer. If the conventional dialogue layer does not have rules that can process the utterance it is analysed to see if it is a

question, if so the utterance is passed to the Oracular layer, if not it is passed to the Aliza layer.
Adam has one simulated emotion, irritation. This can build up or dissipate over a number of dialogue transactions. If a certain level is reached Adam will terminate the conversation. This can happen very quickly (in the case of extremely offensive language from the user) or more gradually with warnings (in the case of the user failing to co-operate with Adam's conversational strategy).
Having illustrated the power of the GO-CA, the following section describes a methodology for developing them.

Section 4 Design methodology for GO-CAs

The software development methodology for the GO-CA combines elements of the staged approach used in the Waterfall model with elements of prototyping or iterative development. The major stages are shown in figure 2.

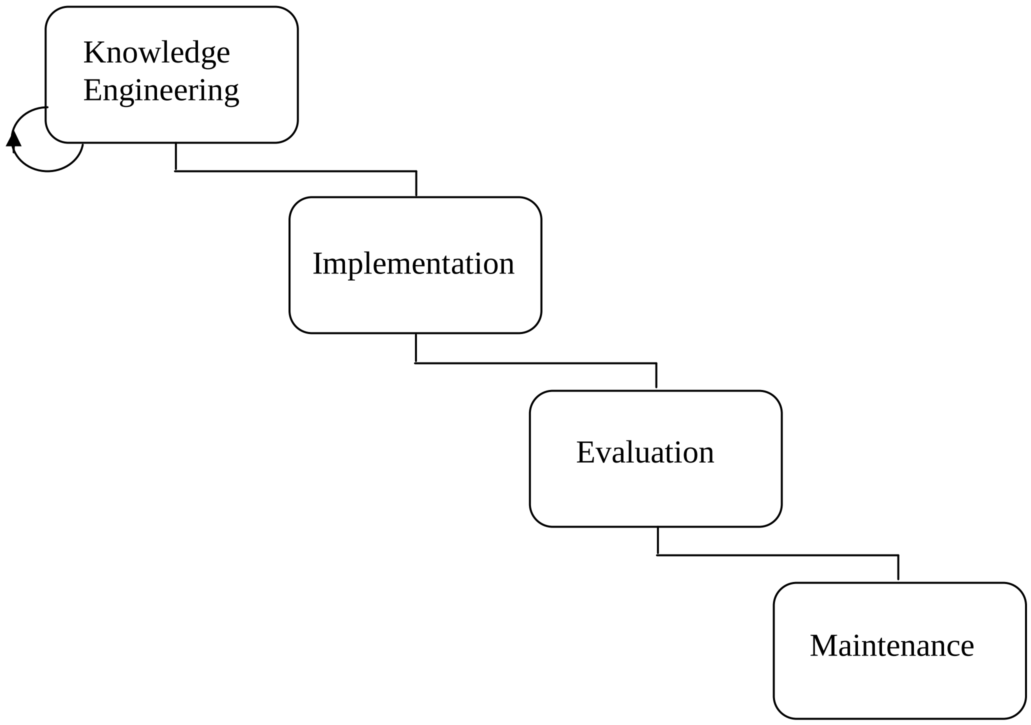

Fig. 2. Design Methodology for GO-CAs

4.1 Knowledge Engineering

Knowledge usually extracts information about a domain from many different sources, including:

- Managers in the client organisation
- Practitioners in the client organisation who interact with the customers who will use the CA being developed
- Documented procedures of the client organisation (e.g. workflow charts)
- 3rd party websites (e.g. government legislation concerning the domain)
- Telephone logs of customer calls related to the domain.

Many organisations do not have formalised processes, instead custom and practice is handed from one generation of employees to another in an oral tradition. This can lead to a reliance on a small number of key individuals in an organisation. Where an organisation has a formalised process for the problem domain, this may not be followed in practice for a variety of reasons e.g. experience and gut feelings, reluctance to change or simply lack of knowledge. So a highly important aspect of knowledge engineering is to formalise the process in the first case or to establish exactly what process should be followed in the second.

To achieve this, a series of meetings is held with key personnel in the organisation who are most knowledgeable about the process and if appropriate, some meetings with the organisation's customers are held.

The processes are refined and the problem domain is expressed in a structured way such as a rule-based system. Figure 3 shows a section of a decision tree rule-based system for bullying and harassment.

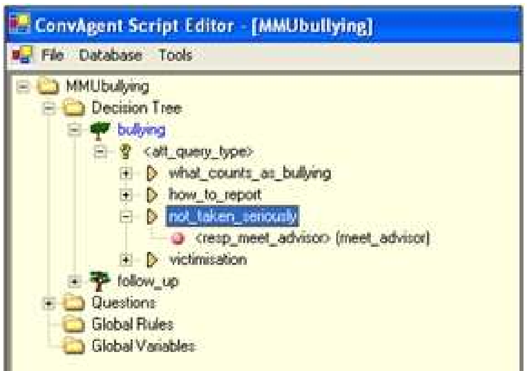

Fig. 3. Bullying and harassment decision tree example

The constructed rule-based system will be validated using a walk-through process and then approved as a true representation of the domain before it is implemented.

When the rule-based system is implemented as an executable program, it is interfaced with the DM and provides the logic for the CA.

A copy of the knowledge base, which includes the rules and other knowledge acquired during the process will be passed on to the scripters, for example particular terminology used in the domain.
The rule base provides information, in terms of its size and complexity, about the overall effort required to construct the CA and is a useful recourse for cost and effort estimation (similar to function point analysis).

4.2 Implementation

All of the components in figure 1 require implementation effort to varying degrees. The user interface is an area where substantial quantities of library code may already exist. In this case effort is restricted to relatively simple tasks in laying out the radio buttons, text boxes etc. Likewise if the application requires a database it is likely to exist already within the organisation (e.g. personnel database), in which case most of the effort involves interfacing it to the system. The logic processing engine may also be able to benefit from previously existing code, for example calculating the number of days expired between two calendar dates is quite a common task.
The rule-based system is the product of the knowledge engineering stage described previously. This is often implemented as high-level code executed on a generic engine. The engine is likely to be a stable piece of code which is re-usable between applications. Rarely, an application may introduce some new demand which requires an engine modification.
Each application will require a customised DM. The DM is the glue that binds the other components of the CA together. Its principal function is to maintain the list of attributes required by the rule-based system. As each new attribute is requested by the system, the DM takes control, continually re-entering the DA until a transaction between the agent and the user captures the required attribute. So the DM is unique to each application, but is relatively small in size.
The bulk of development effort is spent on the DA. The DA is implemented using a scripting language which executes on an interpreting engine.

4.3 Scripting Language

Most scripts use the pattern matching technique, which has been identified as one of the most common and capable methods for developing dialogues that seem to be coherent and intelligent to users [6]. One of the most influential pattern matching approaches was published widely during the 1990s [53] in “The Zen of Scripting Verbots.” This stimulated the production of a number of scripting languages [54] in use today.
A script file consists of rules each of which contains a number of patterns and a response, executed by an engine, for example the DA in figure 1. A user utterance is compared with the patterns in a rule and a numerical activation is calculated (the better the match, the higher the activation). When all of the rules have been processed, the rule with the highest activation fires and its response is used to reply to the user. A threshold can be set for the activation, below which a rule can not fire. If no rule exceeds the activation a default rule will recover the thread of discussion. If useful

information is generated as a result of the rule firing it is passed to other programs making up the agent for relevant action.

The rules will often be divided into contexts to make them more manageable, corresponding to modularisation of conventional code. An application may require many script files covering the various conversational contexts that may occur and there are also mechanisms for switching contexts.

Every time the user types an utterance, every pattern in every rule in the current context must be tested and each test requires multiple passes through the utterance and the pattern. If a user utterance fires a rule that switches to a new context, then the whole process may have to be repeated until there is a match. Pattern matching is a computationally intensive process and producing scalable CAs depends on skilled context design by the people who write the scripts as well as the software engineers who create the engines. The technique is illustrated by the following example, adapted from Plantec [53]:

Consider a pair of activation-based rules:

```
<what-work>
a:0.5
p:60 What *your*job*
p:60 How *earn* living*
r:I’m a full time Verbot
+:<explain>

<explain>
a:0.5
p:60 What *you* mean*
p:60 *Eh*
p:60 *explain*
r: I am a computer program that chats with you.
```

Suppose the first user utterance is

```
 “What is your job?”
```

The engine will begin by comparing it with the first pattern in the first rule:
What *your* job*
In brief the “What” sections of both strings match, but the following “is” and “your” do not. However the wildcard * is able to absorb the “is” and matching continues for the “your” substring. Wildcards are allowed to match nothing so the terminal * on “your” is ignored and the substring “job” matches. Finally the “?” symbol on the end of “job” in the utterance is absorbed by the terminal * in the pattern. This is a match which generates a positive numerical score. This score also depends on how close the match between the utterance and the pattern is (for example how much of the utterance the wildcards have to absorb).

The second pattern in the first rule is then tested and it falls at the first post, because the “H” in How fails to match the “W” in what and there are no leading wildcards to accommodate the difference.
When the second rule is processed the first pattern will begin by matching but will fail at the point where "job" is compared to "mean." The process continues for the remaining patterns.
So the first rule fires (wins) and the agent will reply

```
"I'm a full time Verbot."
```

At this point the user will make another utterance. If there were no matches with contextually meaningful rules, the script could fall back on a general rule that would reply with something like:

```
"What on earth you mean by that?"
"Eh?”
```
or
```
"Could you explain that for me please?"
```

In a large base of rules the corresponding patterns (or variants on them) could occur many times so a promotion mechanism is used to ensure that the correct rule fires.
The entry +:<explain> at the end of the first rule temporarily boosts the activation for the rule <explain> for the next few utterances that the user makes. This feature is known as promotion and the complementary process which temporarily cuts the activation, -: is known as demotion. Various decay functions can be set to govern the return to the original activation over successive utterances in the conversation. This gives some feel for the complexity that can be involved in debugging a large CA.
Other features of scripting include the large range of tuneable parameters for example:
a:0.5 sets a base activation value for the rule, which is principally used to allow one of the rules to fire when matched.
p:60 sets an activation strength for an individual pattern; this is used to prioritise patterns within a rule or instances of a pattern when it appears several times in different rules.
Part of the success of pattern matching may be attributed to its abilities in feature extraction and forming associations. Two particular mechanisms allow it to mimic (to a limited extent) properties of human consciousness. The first is the promotion / demotion of selected rules. This simulates the “stream of consciousness” in which a particular thought comes into the foreground of consciousness against a background of other partially-activated thoughts and ideas [55]. The second is the organisation of rules into contexts (so named because they correspond to a particular conversational context). This allows the agent to focus its attention on a particular topic and prevents misfiring of rules that would correspond to a human being distracted. For example, if a user introduces the name Pluto into a conversation the context could be astronomy or it could be Mickey Mouse’s dog. If each of these topics has a corresponding context, when one is being executed by the DA the rules in the other (and all other contexts) are inaccessible.

Creating scripts is a highly skilled craft [49], requiring the anticipation of user utterances, generation of permutations of the utterances and generalisation of patterns through the replacement of selected terms by wild cards. Modifications to rules containing the patterns can impact on the performance of other rules and modern pattern matching systems contain many parameters that further modify their behaviour.
The main strengths of the technique described in this section are:

- It works well within its limits and it's about the only technique that currently works at all for extended dialogues.
- The computational engines for such systems are well-developed and robust; they are rarely crashed by unexpected user input.
- The scripting method separates out language skills from coding skills. People with language skills can become scripters without learning a great deal of computer science.

However, it also suffers from a number of weaknesses, some of which are:

- Writing patterns which match user inputs effectively is a labour intensive process and the scripters must be highly skilled at selecting key words or phrases and integrating them with wildcards.
- The CA's responses to the user must also be crafted to maintain the dialogue along predictable lines. Transactions which are plausible in isolation can be stilted or incoherent as a complete conversation.
- The organisation of rules into coherent contexts involves another set of skills, similar to the design of coherent modules in conventional programming. Failure to do this results in systems that are difficult to test and debug. They are also easily destabilised by the addition of a single rule.

These drawbacks have an impact on development costs, maintainability and scalability.
However, on the important issue of scalability, pattern matching systems do not require pre-processing stages such as stemming, pos-tagging, syntactical repair and parsing. This is an important consideration as a real-world system could have millions of instances of the CA running simultaneously on the organisation's servers.

4.4 Evaluation

It is possible to evaluate the rule-based system before the rest of the CA is constructed. This is performed using a technique known as "Wizard of Oz" in which a human simulates the CA interface and operates the rule-based system [56]. Users believe they are talking to a real CA because the wizard is hidden by the interface. All of the "agent's" dialogue is provided by the wizard, who extracts the attributes for the rule-based system.
A substantial amount of work has been done on evaluating agents as a whole. The seminal work in this area was the creation of the PARADISE framework [57] which

was applied to evaluate the DARPA communicator SDS [58]. An important feature of PARADISE is the application of linear regression for deriving abstract, indirect attributes such as User Satisfaction in terms of directly measurable attributes [59].
The PARADISE framework continues to provide the framework for current evaluation of CAs including further development of evaluation methodologies [60], CAs for navigation [61], dialogue management strategies [62], tutoring [56] (in press), human-robot dialogue [63] and companion agents [64].
The metrics used to evaluate CAs can be broken down into 3 categories:

- Aspirational Subjective Measures
- Attempted Subjective Measures
- Objective Measures

Prima facie, the subjective measures are important, but they are also difficult to measure with the scientific rigour one might expect of a physical variable such as voltage.

4.4.1 Aspirational Subjective Measures

A number of publications discuss very high level, abstract and subjective concepts which would be very difficult to measure as a single attribute. The most common attributes are:

- Usability [40, 56, 65-69]
- User satisfaction [56, 65, 68, 70-72]
- Agent credibility [70, 73, 74]

The first two are common but difficult to measure attributes from the field of software engineering [59]. There are many more intangible and vague attributes mentioned in studies, including: "Fun to talk with" [38], "lovely, pleasant, black humorous" [70], "Intimacy, Benevolence" [37], "Comfort, Solidarity, Familiarity" [75] and "Trust, Uncertainty, Attractive" [66].

4.4.2 Attempted Subjective Measures

Some studies then go on to attempt to measure a subset of subjective attributes. These are largely measured using Likert or Likert-like attitude rating scales. Attributes measured in this way include:

- Ease of use / Task ease [56, 58, 61, 63, 69, 76-78]
- Ease of the user understanding the agent [56, 58, 63, 66, 76, 77]
- The agent's understanding of the user comprehension [56, 65, 66, 77]
- Various cognitive attributes related to comprehension and complexity [56, 61, 65, 67, 73, 76]

- Various attributes related to the reliability of the agent and the ease of correcting misunderstandings [65, 76]
- Various attributes concerning the user's expertise (of the domain or using the agent) [58, 66, 67, 77]
- The efficiency or effectiveness of the agent [56, 63, 65-67, 76, 78]
- Various attributes about command and control of the conversation [65, 73, 76] [66, 67]
- The pace of the interaction [76, 77]
- Whether the agent behaved as expected [56, 58, 61, 64, 65, 77]
- How natural the agent's behaviour seemed [61, 73, 75, 78]
- Various positive emotional attributes (e.g. friendliness, enjoyment) [65, 66] [63, 75, 76]
- Various negative emotional attributes (e.g. boredom, fluster) [63, 65, 76]
- Whether the user would use again [58, 65, 76, 77] or prefer human service [56, 61, 76]

There are also a substantial number of attributes which occur once or twice including "like further help" [65], "narrative skills" [70], "needs improvement" [76], "question answering capability" [70] and "how much willing to pay" [37].

4.4.3 Objective measures

Most studies include a set of objective measures. Generally speaking, there is a leap of faith that these in some way reflect the aspirational subjective measures that appear at the beginning of published studies. The only systematic and scientific approach was that taken by the PARADISE framework [58]. Attributes measured in this way include:

- Dialogue / Conversation length [38, 40, 58, 61, 63, 65, 72, 73]
- Count of dialogue turns [9, 40, 58, 61, 63, 72, 77-79]
- Various measures of success at utterance or task completion level [58, 61, 63, 65, 68, 72, 79]
- Various counts of errors, corrections or percentage error rates [6, 38, 61, 63, 64, 71, 77, 79]
- Various counts of correct actions by the agent (e.g. answering questions) [58, 66, 67, 70]
- Various speech recognition accuracy measures [9, 74, 76]

Again there are a substantial number of attributes which occur once or twice including "mental workload" [72], "learning gains" (in a tutoring system) [9], count of help messages [76], percentage of time user spent looking at (embodied) agent [67] and user trust of agent (using a standardised measure from psychology) [37].
All recent work makes use of some of the fundamental PARADISE measures whilst adding some application-specific elements. For example, companion agents add more emotional evaluation including the nature of the relationship between the user and the

companion, and the appearance, personality, emotion demonstrated and social attitudes of the companion [64].

4.5 Maintenance

CAs require maintenance in the same manner as any other form of software, however their nature dictates that at the current state of the art they require it to a greater degree. There are 3 drivers of the maintenance requirement, these may be termed:

- Conventional bugs
- The user paradigm shift
- Domain stability

4.5.1 Conventional bugs

Conventional bugs are the same as those in other software; these may be syntax errors or logic errors. They can occur in the three classes of software making up the agent, the rule-based system, the dialogue scripts and the engines. Most errors in the rule-based system will be discovered and removed during the latter stages of the Knowledge Engineering process. However, as with any software development process some will get through.

Once a development team has a set of established engines (such as the DA in figure 1), they will become a relatively infrequent source of bugs and the usual pattern is established of bugs arising largely as a matter of upgrades to the functionality of an engine. As the dialogue scripts are specifically written for each application they differ from previous code and are more likely to provide a source of bugs. Detecting and correcting conventional bugs is a very familiar process for systems engineers, consequently the other two classes pose the greater challenge for CA developers.

4.5.2 The user paradigm shift

Knowledge engineering is an excellent way of capturing existing processes for the production of a CA. However, it does capture the *status quo*, in which user behaviours are shaped by the social interaction with a human expert and the understanding that the expert will have specific expectations of the conversation. However, when faced with a web-based CA the human users behave differently.

One example is the student debt advice system developed [31]. This system was designed to steer students through the process of obtaining the money to pay off their debts to the university by chasing late payments from the student loan company, seeking alternative sources of access funding etc. However, after the system was deployed on the web, logs showed a significant number of students accessing it before starting university to find ways to avoid getting into debt.

This immediately faces the maintainer with decisions, whether to cater for the changed demands from the users, if so to what degree, and how much cost and effort can be justified for implementation.

Considering the goal-oriented architecture, a limited change could be accommodated by adding to the scripts executed by the DM whereas a more extensive change would also require the creation of new attributes processed by the rule-based system. Changing both creates more maintenance effort (modifications to the DM would also be required to communicate the attributes between the DM and rule-based system). Consequently the temptation is to limit the changes to rules in the scripts, but relying too much on scripted rules without the discipline of the rule-based system leads to unstable behaviour of the kind exhibited by poorly-performing chatterbots.

4.5.3 Domain stability

By far the most serious problems arise from domain instability. Domain instability refers to the rate of change in the environment in which the CA operates. There are a number of sources of domain instability. The client commissioning an agent operates in some form of market. When the agent is completed and released into the market all may be well, but markets change over time affecting the relevance of the agent. Another very important source of change is legislation. Many agents will be required to comply with (or explain) requirements of government legislation. Unless the agent is kept up-to-date again it will become less relevant and will generate fewer satisfactory outcomes from the dialogues.

One particular example of this is the UK student loan system which was fundamental to the student debt advisor. This system has changed virtually every year following changes of policy and changes of government.

How may these maintenance challenges be met? One way to tackle the volume of work in maintaining pattern matching systems is to improve the efficiency of the laborious process of hand-crafting the rules in the scripts. Tools have been developed [31] to automate pattern generation and process standardised dialogue templates by a fill-in-the-blanks approach. Other possible tools under consideration include a conflict detector to find conflicts between similar patterns in different rules. This would solve the common problem of adding new rules which overlap with existing rules and conflict with them in firing. Also, current testing requires manually typing utterances into the agent and checking the responses. A regression testing tool containing its own scripts of user utterances with expected outputs from the agents could make this process more efficient. Again regression testing is important to ensure that changes to an agent have not introduced new problems. Test data for regression testing may be accumulated from logs of earlier user evaluations or from conversational scenarios created to design the contexts.

The second approach is to develop an entirely new method of matching user utterances, which does not require the generation of large numbers of patterns. One promising technique, Short Text Semantic Similarity is described in the following section.

Section 5 Novel Algorithms – Short Text Semantic Similarity

The potential for Short Text Semantic Similarity (STSS) algorithms to improve CAs arises from their replacement of the pattern matching component. Suppose an STSS algorithm produces a numerical measure of semantic similarity, this could be used to make judgments such as:

- A pair of STs is identical in meaning
- A pair of STs is completely unrelated in meaning
- One pair of STs is more similar in meaning than another.

Consequently an incoming user utterance could be compared with a number of prototype statements from the domain and an appropriate action and response chosen based on the value of the best match. Consider the following patterns, taken from a rule in a student debt advisor system:

```
p:15 *can*not *afford *pay*
p:15 *can*not *afford *full amount*
p:15 *<problem>* pay*
p:15 . . . . . .. (many more)
```

These patterns will match utterances such as:

I cannot afford to pay you anything this term
I can’t afford the full amount but I could manage to pay a third
There is a difficulty in paying because I was mugged
(amongst many others).

It is clear that even with wildcards for generalisation, many patterns will be needed for good coverage of the overall conversational space. Also there will be a need for skilled scripters who can anticipate user utterances, generate permutations of the utterances, reduce these permutations through generalisation to patterns (use of wild cards) and, very importantly, anticipate interactions between rules.
The alternative offered by STSS is to build the rules from a set of prototype or archetype STs. Suppose, instead of patterns; we had rules containing the following STs:

I can not afford to pay. (ps1)
My money has not come from the Student Loan Company. (ps2)

The user utterance

I cannot afford to pay you anything this term. (u)

would be compared with all of the prototype statements using the STSS algorithms and the highest similarity match would win, as expressed in equation 1:

$$\mathrm{sim}(ps1,u) > \mathrm{sim}(ps2,u) \tag{1}$$

The rule containing ps1 would win and the action specified for the rule would be taken (an attribute set, response to user generated etc.)
Early work on text similarity was concerned with relatively long documents. Consequently similarity was measured using exact matches between words in relatively long vectors of words selected from their respective documents based on their significance [80, 81] Utterances in dialogue are much shorter, and two utterances which convey largely the same meaning may share no common words at all.

5.1 The STASIS algorithm

The STASIS algorithm [82] was specifically designed to overcome this problem. Its key features are:

- Short vectors derived only from the words in the STs
- Use of function words, specific word forms (no stemming/lemmatisation)
- Exploitation of word order information.

Function words are high-frequency closed-class words e.g. articles and auxiliary verbs. In the two sentences "Could you pass the salt?" and "Did you pass the salt?" a single word changes the speech act [83], the overt meaning and the subtle implications of the basic propositional content.
Following the larger-scale publication of STASIS [82], there has been a flurry of work in the STSS field. The majority of subsequent work in the field is either derivative from or influenced by STASIS [84-94]. Accordingly, the following description of STASIS should prove useful to Systems Engineers wishing to develop STSS algorithms.
STASIS uses two stages to calculate the overall semantic similarity between two Short Texts: construction of two vectors (semantic and word-order), followed by combination of the similarity information obtained by the vectors. This is shown in figure 4.
In the following, taken from [82], the lexical database, corpus and word similarity measure components can be replaced by alternatives, although the word similarity measure used in [95] is recommended.

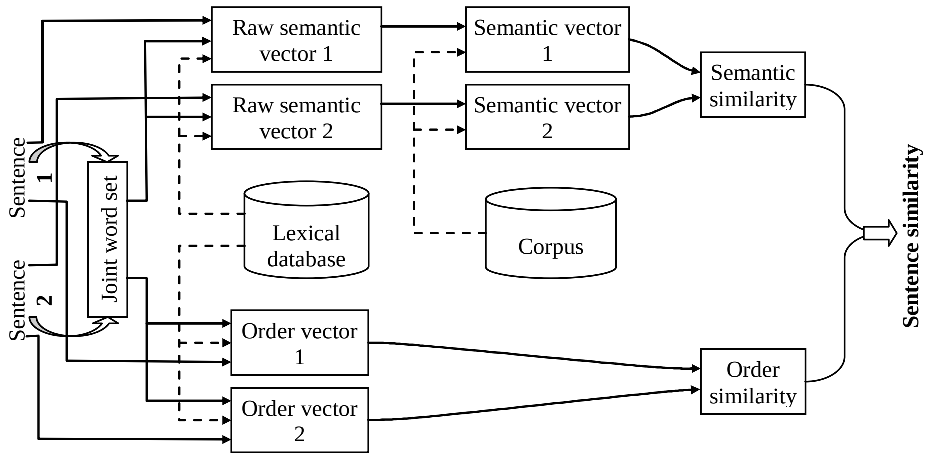

Fig. 4. STASIS sentence similarity computation diagram

In summary the semantic similarity is calculated as follows:

- A joint word set T is derived from all of the distinct words in two short texts, T1 and T2 (equation 2).

$$T = T_1 \cup T_2 = \{w_1 \quad w_2 \quad \cdots \quad w_m\} \tag{2}$$

- A lexical semantic vector $\breve{s}$ is derived from the joint word set for each short text. Each entry, $\breve{s}_i (i=1,2,...,m)$, is determined by the semantic similarity of the corresponding word in the joint word set to a word in the short text (where *m* equals the number of words in the joint word set). Semantic similarity between non-identical words is calculated using the Wordnet ontology.

The words are weighted according to their information content [96] using equation 3:

$$s_i = \breve{s} \cdot I(w_i) \cdot I(\tilde{w}_i) \tag{3}$$

where $I(w_i)$ is the information content of a word in the joint word set and $I(\tilde{w}_i)$ is the information content of its associated word in the short text.

- The semantic similarity (S_s) between the two short texts is calculated using a cosine-like measure between the semantic vectors s_1 and s_2 using a cosine-like function (equation 4):

$$S_s = \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} \tag{4}$$

Word order similarity is calculated as follows:

- Word order vectors, $\mathbf{r_1}$ and $\mathbf{r_2}$ are constructed using the index numbers of words from the joint word set to represent the words in each of the short texts.

- The word order similarity component (S_r) is calculated as the normalised difference in word order (equation 5):

$$S_r = 1 - \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \tag{5}$$

Finally short text similarity is calculated using a weighted sum of the two components using equation 6:

$$S(T_1, T_2) = \delta \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} + (1 - \delta) \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \tag{6}$$

The parameter δ (which adjusts the relative contributions of semantic and word order) is in the range $0.5 < \delta < 1$ and was chosen empirically. Some preliminary experiments have shown potential for significant improvement in performance by optimising δ through linear regression or by combining the two components using a Neural Network [97]. Recent work on embedding the STASIS algorithm in a CA has shown considerable promise [92].

5.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) offers an alternative approach to STSS measurement [98]. Although it was originally designed for document retrieval, the LSA website has a front-end for calculating the similarity between pairs of sentences [99]. There are a number of implementations available on the web. A full description of LSA can be found in [98].
The first stage of LSA is the construction of a semantic space as follows:

- The collection of documents comprising the space is pre-processed to remove all but the most useful terms
- A term-by-document matrix is constructed (X)
- This large matrix is decomposed by Singular Value Decomposition into 3 other matrices, one derived from the terms (T_0), one from the documents (D_0) and a diagonal matrix (S_0) linking them
- The reduction in the size of S_0, which reduces the size of T_0 and D_0

- An approximation of the original space, $\hat{X}$, is reconstructed according to equation 7:

$$X \approx \hat{X} = TSD^{'} \qquad (7)$$

LSA generalises by spreading information from a cell to its related cells; whereas *X* contains a small number of 1s and a large number of 0s, $\hat{X}$ will contain many more non-zero values. Thus when a text is projected into the semantic space, more cells (and therefore spatial dimensions) contribute to its representation, enabling LSA to calculate a similarity even when no co-occurrence exists between particular terms. Short text similarity is calculated as follows:

- A row in the semantic space is formed for each ST
- If a text did not appear in the original space it is constructed as a pseudo object using equation 8:

$$D_q = X'_q T S^{-1} \qquad (8)$$

The method for this is disclosed in [100].

- Similarity is measured as the angle between the vectors representing the two short texts using equation 9:

$$\hat{X}\hat{X}^{'} = TS^2T^{'} \qquad (9)$$

The performance of LSA depends on the semantic space. It is possible to use the best performing semantic space from LSA's creators [98], in which case only the steps for similarity calculation are required.

Section 6 Research opportunities

There are opportunities for systems engineers to perform research on virtually every aspect of CAs. Starting with SDS, the key problem is the vocabulary. The ultimate research aim must be a universal SDS interface that allows CAs to be deployed over the internet for the general population. However, the state of the art is a high WER with large vocabulary systems or a small vocabulary with more reliable systems [14, 16]. Potential lines of research are:

- The production of multi-classifier systems adding more diverse elements such as neural networks to the current statistically-based techniques
- Development of generic (rather than customised) medium-sized vocabularies that would be re-usable across different projects in the same general domain
- Development of an open SDS protocol that locates the speech recognition in the user's mobile phone (such a system would be trained only once and

would act as the SDS front end for any application the phone owner wished to use).

Chatterbot systems still offer a useful starting point for systems engineers to become involved in CA research or development. In particular, the ALICE AIML chatterbot technology is readily available and well documented. Furthermore, ALICE also illustrates that there is potential for introducing new creative techniques to improve chatterbots: the symbolic reduction (SRAI) instruction allows an AIML chatterbot to re-enter itself recursively to decompose complex user utterances. Potential lines of chatterbot research are:

- Development of software tools to improve productivity of scripting
- Development of tools to support debugging and maintenance
- The extension of chatterbot engine functionality.

The chief problem of Natural Language Processing is that CA development has a minor role in the field as a whole, with its main interests being in areas such as information retrieval and machine translation. Undoubtedly many interesting and novel algorithms, architectures and processes have been developed, but in a piecemeal fashion. Therefore the research challenges for NLP systems are:

- The construction and evaluation of real-world systems that can be deployed for use by the general public
- Measurement of the scalability of such systems to realistic numbers of users in real-time.

GO-CAs are a recent development and opportunities for research are plentiful. Some lines include:

- Development of alternative top-level component architectures (for example through the use of alternative intelligent components such as neural networks to drive the goals)
- Development of alternative methods for the DA to communicate with the other components (at present the DA is a pattern matching engine)
- Application of new algorithms such as STSS within the DA
- Development of new knowledge engineering tools and processes
- Development of authoring and maintenance tools

ECAs present the same kinds of opportunities for systems engineers as NLP systems. ECA research interest has shifted from fairly mundane but obviously useful topics such as disambiguation through pointing at objects and co-operative use of objects such as maps in shared visual space, to more abstract topics like measuring engagement in multi-party dialogues and simulation of emotions. Again the immediate research topics of interest to a systems engineer should be:

- Development of ECAs using current properties that are believed to be useful (e.g. pointing, emphatic gestures etc.)
- Applying such ECAs to realistic problems requiring larger numbers of complex attributes
- Objective evaluation of the gain (in terms of metrics such as successful task completion, length of dialogue etc.) obtained from ECA features.

Section 7 Conclusions

CA technology is something that all systems engineers should be aware of. GO-CA technology has reached the point where it is possible to build and deploy real-world applications and some systems engineers in industry will find that this forms part of projects they will work on during their careers. How far CAs penetrate into mainstream computing will depend on research in the short and medium term. This research, particularly involving development of authoring and maintenance tools, and the objective evaluation of tools, algorithms and techniques, will be crucially dependent on the skills of the systems engineer to be successful.

Glossary

ALICE:	Artificial Linguistic Internet Computer Entity
CA:	Conversational Agent
DA:	Dialogue Agent
DARPA:	Defense Advanced Research Projects Agency
DM:	Dialogue Manager
ECA:	Embodied Conversational Agent
GO-CA:	Goal Oriented Conversational Agent
LSA:	Latent Semantic Analysis
NLP:	Natural Language Processing
NLU:	Natural Language Understanding
PARADISE:	a framework for evaluating and comparing the performance of spoken-language dialogue systems
REA:	a Real Estate Agent
SCHISMA:	SCHouwburg Informatie Systeem (a Dutch theatre booking system)
SDS:	Spoken Dialogue System
SMS:	Short Message Service (texting)
SRAI:	Symbolic Reduction Artificial Intelligence
STASIS:	a specific instance of an STSS algorithm
STSS:	Short Text Semantic Similarity
VSA:	Virtual Seller Agent
WER:	Word Error Rate

References

1. Turing, A.M., *Computing Machinery and Intelligence.* Mind, New Series, 1950. **59**(236): p. 433-460.
2. Gunderson, K., *The Imitation Game.* Mind, New Series, 1964. **73**(290): p. 234-245.
3. Searle, J.R., *Minds, brains and programs.* Behavioural and Brain Sciences, 1980. **3**: p. 417-424.
4. Block, N., *Psychologism and behaviourism.* The Philosophical Review, 1981. **LXXXX**(1): p. 5-43.
5. Searle, J.R., *Mind, Language and Society*. 1999: Weidenfield & Nicholson
6. Bickmore, T. and T. Giorgino, *Health dialog systems for patients and consumers.* J Biomed Inform, 2006. **39**(5): p. 556-571.
7. Cassell, J., et al., *Embodied CAs*. 2000.
8. Gorin, A.L., G. Riccardi, and J.H. Wright, *How may I help you?* Speech Communication, 1997. **23**: p. 113-127.
9. Graesser, A.C., et al., *AutoTutor: An Intelligent Tutoring System With Mixed Initiative Dialogue.* IEEE Transactions on Education, 2005. **48**(4): p. 612-618.
10. Owda, M., Z. Bandar, and K. Crockett. *Conversation-Based Natural Language Interface to Relational Databases*. in *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*. 2007.
11. Glass, J., et al. *A Framework for Developing Conversational User Interfaces*. in *Fourth International Conference on Computer-Aided Design of User Interfaces*. 2004. Funchal, Isle of Madeira, Portugal: p.347-358.
12. McGeary, Z., et al., *Online Self-service: The Slow Road to Search Effectiveness*, in *Customer Relationship Management*. 2005.
13. Hunt, M.J., *Figures of Merit for Assessing Connected Word Recognisers* Speech Communication, 1990. **9**: p. 239-336.
14. Hosom, J.-P. *Automatic Speech Recognition at CSLU*. 2003 [cited 20/10/2010]; Available from: http://cslu.cse.ogi.edu/asr/.
15. Hunt, A. *comp.speech FAQ Section 6*. 1997 [cited 20/10/2010]; Available from: http://www.speech.cs.cmu.edu/comp.speech/Section6/Q6.1.html.
16. Raut, C.K., *Discriminative Adaptive Training and Bayesian Inference for Speech Recognition*, in *Emmanuel College*. 2009, University of Cambridge.
17. Hillard, D.L., *Automatic Sentence Structure Annotation for Spoken Language Processing*, in *Electrical Engineering*. 2008, University of Washington.
18. Zdenek, S., *Passing Loebner's Turing test: A case of Conflicting Discourse Functions.* Minds and Machines, 2001. **11**: p. 53-76.
19. Jackson, P. and I. Moulinier, *Natural Language Processing for Online Applications*. 2 ed. Natural Language Processing. 2007, Amsterdam, The Netherlands: John Benjamins Publishing Company.
20. Zdravkova, K., *Conceptual Framework for an Intelligent and ChatterBot*, in *22nd International Conference Information Technology Interfaces ITI 2000*. 2000: p. 189-194.

21. Minker, W., S. Bennacef, and J.-L. Gauvain. *A stochastic case frame approach for natural language understanding* in *Fourth International Conference on Spoken Language, ICSLP 96.* . 1996. Philadelphia, PA: p. 1013-1016.
22. Farquhar, A., R. Fikes, and J. Rice, *The Ontolingua Server: a Tool for Collaborative Ontology Construction.* Journal of Human-Computer Studies, 1997. **46:** p. 707-728
23. Sagae, K., et al. *Towards Natural Language Understanding of Partial Speech Recognition Results in Dialogue Systems* in *NAACL HLT 2009*. 2009. Boulder, Colorado,: Association for Computational Linguistics: p. 53-56.
24. Young, S., et al., *The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management.* Computer Speech and Language, Special Issue on Evaluation., 2010. **24**(2): p. 150-174.
25. Lefevre, F., et al. *k-Nearest Neighbor Monte-Carlo Control Algorithm for POMDP-Based Dialogue Systems*. in *The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 2009. London, UK: p. 272-275.
26. Bohus, D. and E. Horvitz. *Learning to Predict Engagement with a Spoken Dialog System in Open-World Settings*. in *The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 2009. London, UK.
27. Bohus, D. and E. Horvitz. *Models for Multiparty Engagement in Open-World Dialog* in *SIGDIAL 2009: the 10th Annual Meeting of the Special Interest Group in Discourse and Dialogue*. 2009. Queen Mary University of London: p. 225-234.
28. DeVault, D., K. Sagae, and D. Traum. *Can I finish? Learning when to respond to incremental interpretation results in interactive dialogue* in *The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 2009. London, UK: p. 11-20
29. Skantze, G. and J. Gustafson. *Attention and Interaction Control in a Human-Human-Computer Dialogue Setting*. in *The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 2009. London, UK: p. 310-313.
30. op den Akker, H. and R. op den Akker. *Are You Being Addressed? - real-time addressee detection to support remote participants in hybrid meetings*. in *The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 2009. London, UK: p.21-28.
31. Crockett, K., et al., *Bullying and Debt: Developing Novel Applications of Dialogue Systems*, in *Knowledge and Reasoning in Practical Dialogue Systems (IJCAI)*. 2009, IJCAI: Pasadena, CA: p. 1-9.
32. Cassell, J., et al., *More Than Just a Pretty Face: Conversational Protocols and the Affordances of Embodiment.* Knowledge-Based Systems, 2001. **14**: p. 55-64.
33. Robinson, S., et al. *What would you ask a CA? Observations of Human-Agent Dialogues in a Museum Setting*. in *Language Resources and Evaluation Conference 2008*. 2008. Marrakech, Morocco: p.1125-1131.

34. Babu, S., et al., *"What Would You Like to Talk About?" An Evaluation of Social Conversations with a Virtual Receptionist.* Lecture Notes in Computer Science 2006. **4133/2006**: p.169-180.

35. Lance, B.J. and S.C. Marsella. *A model of gaze for the purpose of emotional expression in virtual embodied agents* in *International Conference on Autonomous Agents*. 2008. Estoril, Portugal: p. 199-206.

36. Gergle, D., C.P. Rosé, and R.E. Kraut. *Modeling the Impact of Shared Visual Information on Collaborative Reference*. in *The SIGCHI conference on Human factors in computing systems*. 2007: p.1543-1552.

37. Bickmore, T. and J. Cassell. *'How about this weather?' Social Dialog with Embodied CAs*. in *The American Association for Artificial Intelligence (AAAI) Fall Symposium on "Narrative Intelligence"*. 2000. Cape Cod, MA.: p. 4-8.

38. Kopp, S., et al., *A CA as Museum Guide – Design and Evaluation of a Real-World Application* Lecture Notes in Computer Science 2005(3661/2005) : p. 329-343.

39. Bevacqua, E., et al., *An expressive ECA showing complex emotions*, in *AISB'07 - Artificial and Ambient Intelligence*. 2007: Newcastle University, Newcastle upon Tyne, UK.

40. Walker, M.A., L. Hirschman, and J. Aberdeen, *Evaluation for Darpa Communicator Spoken Dialogue Systems* in *Language Resources and Evaluation Conference* 2000: Athens, Greece.

41. Walker, M.A., R. Passonneau, and J.E. Boland. *Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems*. in *The 39th Annual Meeting on Association for Computational Linguistics* 2001. Toulouse, France: p. 515-522.

42. Walker, M.A., et al. *Speech-Plans: Generating Evaluative Responses in Spoken Dialogue*. in *International Conference on Natural Language Generation*. 2002: p. 73-80.

43. Giraudo, E. and P. Baggia. *EVALITA 2009: Loquendo Spoken Dialog System*. in *Evaluation of NLP and Speech Tools for Italian EVALITA'09*. 2009. Reggio Emilia, Italy.

44. Rigo, S., et al., *The 2009 UNITN EVALITA Italian Spoken Dialogue System*, in *Evaluation of NLP and Speech Tools for Italian EVALITA'09*. 2009: Reggio Emilia, Italy.

45. De Angeli, A., *Ethical implications of verbal disinhibition with CAs.* PsychNology Journal, 2009. **7**(1): p. 49-57.

46. Kegel. *Kegel - Oldest Harley Dealer*. [cited 23/3/2010]; Available from: http://kegelmotorcycles.com/.

47. Hulstijn, H., et al., *Topics in schisma dialogues*, in *Twente Workshop on Language Technology 11 (TWLT11)*. 1996: University of Twente.

48. Andernach, T., et al. *Language Analysis for Dialogue Management in a Theatre Information & Booking System*. in *15th International Conference on Language Engineering, AI 95*. 1995. Montpellier: p. 351-362.

49. Michie, D. (2001) *Return of the Imitation Game*. Electronic Transactions in Artificial Intelligence **6(B)**, 205-220

50. BBC. *Tories criticise flu advice line* 2009 06:45 GMT, Friday, 24 July 2009 07:45 UK [cited 24/7/2009]; Available from: http://news.bbc.co.uk/1/hi/health/8166444.stm.
51. Morge, M., S. Abdel-Naby, and B. Beaufils. *Towards a dialectical approach for CAs in selling situations*. in *The 9th International Conference on Autonomous Agents and Multiagent Systems*. 2010. Toronto, Canada: p. 127-144.
52. Lance, B., Marsella, S. *A Model of Gaze for the Purpose of Emotional Expression in Virtual Embodied Agents* in *7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*. 2008. Estoril, Portugal: p.199-206.
53. Plantec, P. *The Zen of Scripting Verbots*. 1998 [cited 2004 28/9/04]; Available from: http://web.archive.org/web/19991013075513/vperson.com/verbotzen30tt.html.
54. Sammut, C. (2001) *Managing Context in a CA*. Electronic Transactions in Artificial Intelligence **5(B)**, 191-201
55. Dehaene, S. and L. Naccache, *Towards a cognitive neuroscience of consciousness: basic evidence and a workspace framework.* Cognition, 2001. **79** p. 1-37.
56. Forbes-Riley, K. and D. Litman, *Designing and evaluating a wizarded uncertainty-adaptive spoken dialogue tutoring system.* Computer Speech and Language, Special Issue on Evaluation, 2011. **25**(1): p. 105-126
57. Walker, M.A., et al. *PARADISE: a framework for evaluating spoken dialogue agents* in *the 35th Annual Meeting of the Association for Computational Linguistics* 1997. Madrid, Spain: p.271-280.
58. Walker, M.A., et al., *Darpa communicator dialog travel planning systems: The june 2000 data collection* in *EUROSPEECH 2001 7th European Conference on Speech Communication and Technology 2nd INTERSPEECH Event* 2001: Aalborg, Denmark: p. 1371-1374.
59. Fenton, N. and S. Pfleeger, *Software Metrics: A Rigorous and Practical Approach.* 1998: PWS
60. Kato, T., M. Matsushita, and N. Kando. *Bridging Evaluations: Inspiration from Dialogue System Research*. in *SIGIR 2010 33rd Annual International ACM SIGIR Conference*. 2010: SIGIR: p.3-4.
61. Dethlefs, N., et al. *Evaluating Task Success in a Dialogue System for Indoor Navigation* in *SemDial 2010 14th Workshop on the Semantics and Pragmatics of Dialogue*. 2010: p.143-146.
62. Lee, C., et al., *Recent Approaches to Dialog Management for Spoken Dialog Systems.* Journal of Computing Science and Engineering, 2010. **4**(1): p. 1-22.
63. Foster, M.E., et al. *Evaluating Description and Reference Strategies in a Cooperative Human-Robot Dialogue System* in *The 21st international jont conference on Artifical intelligence table of contents*. 2009. Pasadena, California, USA: p.1818-1823.
64. Webb, N., et al. *Evaluating Human-Machine Conversation for Appropriateness*. in *The 7th conference on International Language Resources and Evaluation (LREC'10)*. 2010. Valletta, Malta.

65. Bouwman, G., J. Sturm, and L. Boves. *Incorporating confidence measures in the Dutch train timetable information system developed in the ARISE project* in *ICASSP '99*. 1999: p. 493-496.
66. Semeraro, G., et al., *Evaluation and Validation of a CA Embodied in a Bookstore.* Lecture Notes in Computer Science, 2003. **2615**: p. 360-371.
67. Andersen, V., et al., *A methodological approach for designing and evaluating intelligent applications for digital collections.* Applied Artificial Intelligence, 2003. **17**(8-9): p. 745-771.
68. Lamel, L., et al., *User evaluation of the MASK kiosk.* Speech Communication, 2002. **38**(1): p. 131 - 139.
69. Ortony, A., G.L. Clore, and A. Collins, *The Cognitive Structure of Emotions*1990: Cambridge University Press.
70. Yuan , X. and Y.S. Chee, *Design and evaluation of Elva: an embodied tour guide in an interactive virtual art gallery.* Computer Animation and Virtual Worlds, 2005. **16**(2): p. 109 - 119.
71. McKevitt, P., P. D., and Y. Wilks, *Why machines should analyse intention in natural language dialogue.* Int. J. Human-Computer Studies, 1999. **51**: p. 947-989.
72. Le Bigot, L., E. Jamet, and J.-F. Rouet, *Searching information with a natural language dialogue system: a comparison of spoken vs. written modalities.* Applied Ergonomics, 2004. **35**: p. 557–564.
73. Cassell, J. and H. Vilhjálmsson, *Fully Embodied Conversational Avatars: Making Communicative Behaviors Autonomous.* Autonomous Agents and Multi-Agent Systems, 1999. **2**(1): p. 45-64.
74. Massaro, D.W., et al., *Developing and evaluating CAs*, in *Embodied CAs*, J. Cassell, et al., Editors. 2000, MIT Press: Cambridge, MA. p. 286-318.
75. Cassell, J. and T. Bickmore, *Negotiated Collusion: Modeling Social language and its Relationship Effects in Social Agents.* User Modeling and User-Adapted Interaction 2003. **13**: p. 89-132.
76. Lamel, L., et al., *The LIMSI RailTel System: Field trial of a telephone service for rail travel information.* Speech Communication, 1997. **23**(1-2): p. 67-82
77. Litman, D.J. and S. Pan, *Designing and Evaluating an Adaptive Spoken Dialogue System* in *User Modeling and User-Adapted Interaction*, 2002. **12**: p. 111-137.
78. Sanders, G.A. and J. Scholtz, *Measurement and Evaluation of Embodied CAs* in *Embodied CAs*, J. Cassell, et al., Editors. 2000, MIT Press.
79. Bouwman, G. and J. Hulstijn. *Dialog Strategy Redesign with Reliabilty Measures* in *1st Int. Conf. on Language Resources and Evaluation*. 1998. Granada, Spain: p.191-198.
80. Spärck-Jones, K., *A Statistical Interpretation of Term Specificity and its Application in Retrieval* Journal of Documentation, 1972. **vol. 28**: p. 11–21.
81. Salton, G., A. Wong, and C.S. Yang *A Vector Space Model for Automatic Indexing.* Communications of the ACM, 1975. **18**(11): p. 613–620.
82. Li, Y., et al., *Sentence Similarity Based on Semantic Nets and Corpus Statistics.* IEEE Transactions on Knowledge and Data Engineering, 2006. **18**(8): p. 1138-1150.

83. Austin, J.L., *How to do things with Words: The William James Lectures delivered at Harvard University in 1955*. 2 ed, ed. J.O. Urmson. 1975: Harvard University Press.
84. Ferri, F., P. Grifoni, and S. Paolozzi, *An Approach to Multimodal Input Interpretation in Human-Computer Interaction* in *The Nineteenth International Conference on Software Engineering Knowledge Engineering (SEKE'2007)* 2007: Boston, Massachusetts, USA: p. 664-669.
85. Tsatsaronis, G., I. Varlamis, and M. Vazirgiannis, *Text Relatedness Based on a Word Thesaurus* Journal of Artificial Intelligence Research, 2010. **37** p. 1-39.
86. Min, F., L. Wenyin, and W. Chen. *Answer Clustering and Fusion in a User-Interactive QA System* in *Second International Conference on Semantics, Knowledge and Grid*. 2006: p.41.
87. Gacitua-Decar, V. and C. Pahl. *Automatic Business Process Pattern Matching for Enterprise Services Design* in *2009 World Conference on Services - II*. Bangalore, India: p. 111-118.
88. Huang, J.-J., S.-T. Changt, and S.-Y. Hu. *Searching for Answers via Social Networks*. in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*. 2008. Las Vegas, NV: p. 289-293.
89. Capuano, N., et al. *On-Demand Construction of Personalized Learning Experiences Using Semantic Web and Web 2.0 Techniques* in *Ninth IEEE International Conference on Advanced Learning Technologies*. 2009 IEEE Computer Society Washington, DC, USA: p. 484-488.
90. Inkpen, D., *Semantic Similarity Knowledge and its Applications.* Studia Universitatis Babes-Bolyai Informatica, 2007. **LII**(1): p. 11-22.
91. Achananuparp, P., X. Hu, and X. Shen, *The Evaluation of Sentence Similarity Measures.* Lecture Notes in Computer Science, 2008. **5182**: p. 305–316.
92. O'Shea, K., Z. Bandar, and K. Crockett, *Towards a New Generation of CAs Based on Sentence Similarity.* Lecture Notes Electrical Engineering, 2009. **39**: p. 505-514.
93. O’Shea, K., Z. Bandar, and K. Crockett. *A Novel Approach for Constructing CAs using Sentence Similarity Measures* in *Proceedings of the World Congress on Engineering WCE 2008*. 2008. London, U.K: p. 321-326.
94. Liu, X., Y. Zhou, and R. Zheng. *Sentence Similarity based on Dynamic Time Warping* in *International Conference on Semantic Computing, ICSC 2007* 2007. p.250-256.
95. Li, Y., Z. Bandar, and D. McLean, *An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources.* IEEE Transactions on Knowledge and Data Engineering, 2003. **15**(4): p. 871-882.
96. Resnik, P., *Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language.* Journal of Artificial Intelligence Research, 1999. **Vol. 11**: p. 95-130.
97. O'Shea, J., *A Framework for Applying Short Text Semantic Similarity in Goal-Oriented CAs* in *Computing and Mathematics*. 2010, Manchester Metropolitan University: Manchester.

98. Landauer, T.K., P.W. Foltz, and D. Laham, *An Introduction to Latent Semantic Analysis.* Discourse Processes, 1998. **25** p. 259-284.
99. Laham, D. *Latent Semantic Analysis @ CU Boulder*. 1998 [cited 2008 20/01/08; LSA website]. Available from: http://lsa.colorado.edu/.
100. Deerwester, S., et al., *Computer information retrieval using Latent Semantic Structure*, U.S.P. Office, 1989, Bell Communications Research Inc: United States of America.