Knowledge-Based Systems 69 (2014) 64–77

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Microblogging as a mechanism for human–robot interaction

David Bell [b], Theodora Koulouri [b], Stanislao Lauria [b,*], Robert D. Macredie [b], James Sutton [a]

[a] IBM, United States
[b] Brunel University, United Kingdom

ARTICLE INFO

Article history:
Available online 14 May 2014

Keywords:
Twitter
Human–robot interfaces
Architectures for social robotics
Computational intelligence for knowledge acquisition
Social media retrieval

ABSTRACT

This paper presents a novel approach to social data analysis, exploring the use of microblogging to manage interaction between humans and robots, and presenting and evaluating an architecture that extends the use of social networks to connect humans and devices. The approach uses natural language processing – in the form of simple grammar-based techniques – to extract features of interest from textual data retrieved from a microblogging platform in real-time and generate appropriate executable code for the robot. The simple rule-based solution exploits some of the 'natural' constraints imposed by microblogging platforms to manage the potential complexity of the interactions and create bi-directional communication. In order to evaluate the developed system, an analysis of real-time, user-generated social media data is presented. The analysis demonstrates the feasibility of producing programmes from the social media data which lead to executable actions by a front-end application – an approach of immediate relevance to web-based systems, like question–answering engines, personal digital assistants, and smart home/office devices.

1. Introduction

Electronic communication networks permeate many aspects of our daily lives, offering what now approaches anytime, anywhere access to the internet, supporting inter-personal communication in a range of forms, and allowing individuals to freely create and share content through multiple platforms without requiring coding skills. This content consists of vast amounts of natural language data, providing an unprecedented insight into human social behaviour, sentiments, opinions and expertise on a global scale, creating a wealth of new opportunities for Natural Language Processing (NLP) research, with immediate implications for business and commerce. This new territory, however, also poses challenges for NLP, as the analysis of massive, heterogeneous, unstructured and noisy data in real-time is a formidable task for existing approaches.

In response to the new challenges and opportunities, the paper presents an architecture for social data retrieval and analysis using NLP techniques. While recognising the immense value of machine-learning algorithms, it develops the argument that a simpler rule-based solution that leverages some of the 'natural' constraints imposed by the communication medium itself can also yield promising results. The paper uses a robot as a proof-of-concept system to which the architecture is tuned. It includes the analysis of data generated in real-time by users of a social media platform, in order to assess the performance of the approach. At the same time, the particular front-end application allows for a unique use scenario; in which the social media data from the user is not only analysed, but also acted upon, and, most importantly, new data (in the form of system responses) is shared through the same platform. As such, the present paper adds a dimension to social data analysis, using it for event detection as well as event generation and, as a result, supports bi-directional communication. In addition, the field of human–robot interaction is interesting in its own right, and related research focuses on aspects such as intuitive interfaces and user perceptions. So, the paper ultimately aims to present and evaluate an architecture that extends the use of social networks to connect humans and machines.

As argued above, advances in computer and communication technologies have simplified and, at the same time, enriched interactions between users and between users and systems. Robots, on the other hand, are still relatively inaccessible to most people since communication with them commonly requires knowledge of programming languages or training in the use of complex applications.

The most natural way for humans to communicate and interact is, of course, through spontaneous forms of communication such as speech, so it is unsurprising that there has been an increasing interest in translating natural language instructions given by people into actions carried out by machines (see, for example, the IBL [27] and the DiaSpace projects [42]). So far, research based on these human-oriented approaches to human–machine interaction

* Corresponding author.
E-mail addresses: stasha.lauria@brunel.ac.uk (S. Lauria), james@jsutton.co.uk (J. Sutton).

http://dx.doi.org/10.1016/j.knosys.2014.05.009
0950-7051/

has focused on using either speech or graphical techniques to instruct robots to perform tasks. However, speech-based interfaces suffer from inadequate speech recognition accuracy in real-world environments, and graphical control, while valuable in multimodal interfaces, typically requires learning in order to use it. Thus, this paper argues that the pervasiveness of networks and the popularity of specific communication platforms offer opportunities for human–machine interaction at a distance, in near real-time. Specific social networking applications, for example, provide software platforms that use natural language, albeit with some restrictions, and allow the connection in near real-time of individuals and devices (in machine form). This paper will explore the potential of a sub-class of social networking applications – microblogging sites – to be used as a platform for human–machine interaction.

The work reported in the paper will demonstrate how social networks and robotics can be brought together in a way that would allow any naive user to control a robot using simple natural language commands. The simplicity with which messages may be exchanged, combined with the ubiquitous nature of microblogging clients, makes this a communication method that can be seen as a hybrid of instant messaging and status notification and therefore ideal for near real-time applications. The microblogging platform used in the work also means that the exchange can be bi-directional, allowing the robot to message, or enter into dialogue with, the user to clarify any problem that occurs. When seen in combination with the immediacy of the microblogging application, this offers the opportunity for users quickly to become involved in a range of tasks with the robot and confidently to embark on complex activities. In particular, robots can be connected to the wireless home/office network and controlled through handheld devices or computers remotely, in order to perform tasks such as household chores, intelligent teleconferencing, movement and intruder detection, and pet and plant care.

The remainder of the paper is organised as follows. In Section 2, we provide a general discussion of microblogging platforms and their potential in this area, introduce the specific platform used in the work (Twitter), and briefly explore existing research in the area of robotics and social networking. Section 3 describes the general architecture that constitutes the proposed human–machine interaction approach. Section 4 discusses the results of the system evaluation study that involved real users. Finally, Section 5 presents conclusions drawn from the work and identifies areas for future research.

2. Background

In this section, key aspects related to the proposed architecture are presented. In particular, the microblogging service used in the work is briefly discussed and the reasons for its adoption presented. Another element of the approach developed and reported in this paper is the use of social networks in robotics, necessitating a brief background review of relevant work in this area to inform the research activity. This leads to a discussion of work related to event detection tasks, before moving onto event generation tasks, which frames the original aspects investigated in this paper.

2.1. The microblogging platform

Various studies (see, for example, Java et al. [20]) have investigated the uses of microblogging platforms, with coordinating social activities, seeking or sharing information, and reporting news being among the most popular. Analysis of these studies suggests that much of the existing research on textual information processing (or text mining) has been focused on *event detection*. The tasks that have been addressed using text mining from microblog postings include: sentiment analysis; classification of messages into categories; clustering of messages; and identification of trending topics. For example, Petrovic et al. [37] attempt to detect whether users discuss any event that has never appeared before on a microblogging platform (Twitter); and in Sakaki et al. [39], the authors analyse messages on the same platform to detect critical events like earthquakes. In such cases, the researchers formulate event detection as a classification problem. By searching for specific keywords, they have been able to successfully classify tweets (the messages broadcast via Twitter) into a positive or a negative event, showing that limited Natural Language Processing (NLP) techniques can be useful in domain-specific contexts owing to the constraints imposed on, and features offered to, the user by the social media format.

Twitter was chosen as the communication platform used in this research for several reasons. First, it is one of the most commonly-used social networking services on the internet. Since the first 'tweet' – the name for the maximum 140-character messages supported by the platform – was sent in 2006, Twitter has amassed over 200 million active users and, on average, 340 million tweets are sent each day.[1,2] Another key reason for using Twitter in this research is the immediacy of messages sent on the platform. Twitter streams consist of short messages sent to other users in near real-time. Messages may be used to *broadcast* to a wide range of subscribed users (followers) or *narrowcast*, as a one-to-one communication, using a direct message [11]. Twitter is also very low-bandwidth, making it easily accessible across a wide range of networks including those used with mobile/cell phones, and is available across fixed and mobile platforms (personal computers, tablets, smartphones). Twitter is even usable with older mobile phones without web access, since SMS messages sent to Twitter can act as tweets. In addition, Twitter has a simple interface and is basic in its design, making it easy for inexperienced users to gain an understanding of how to use it. This means that anyone with access to a mobile phone or Internet-connected device can use Twitter and could, in the context of our work, communicate with a remote device such as a robot.

Given that Twitter is relatively 'young', there is little scientific data about how it affects language use and how it compares with other (more established) electronic media of communication. Of note, though, is the study by Hu et al. [18] which analysed and compared large language corpora obtained from Twitter, SMS, online chat, emails, Blogs, online magazines and a news website. Compared to SMS and online chat, the language produced over Twitter was found to be more standard, formal and factual. The authors reported that Twitter closely resembles written language norms (such as the ones observed in online magazines and news sites), while sharing the brevity, immediacy and interactivity of SMS and online chat. This evidence challenges expectations that the language data generated on Twitter is highly ungrammatical and idiosyncratic; it, in fact, sets Twitter apart as a unique social media/CMC platform. Based on these findings, it could be argued that these inherent characteristics of language use on Twitter (interactive, constrained but, at the same time, relatively fluent language) combined with Twitter's ubiquity and cross-platform accessibility over the web essentially render Twitter an excellent platform for controlling computers or devices through natural language. In effect, the present study develops the argument that users will naturally *adapt to the platform* and produce simple, concise and clean input for the robot, without the necessity to consciously *adapt to the robot*.

[1] https://business.twitter.com/twitter-101.
[2] http://who.is/website-information/www.twitter.com.

Moreover, Twitter offers a more active and conversational form of communication than other forms of blogging and, as such, it may attract more interest-driven participation. Because previous research suggests that people will treat a robot with the same social conventions that they would a human (see, for example, Kidd and Breazeal [22]), Twitter can then be seen as offering a platform for computer-mediated communication between human and robot. An unresolved question, though, is whether such a computer-mediated medium creates or removes obstacles for successful communication when compared with the face-to-face communication (i.e., direct or 'medium-less' control of robots).

While it has been argued, and empirically demonstrated, that computer-mediation can create obstacles for successful communication in other contexts, there is also a substantial amount of evidence that the impact of the resulting obstacles on media choice and task outcomes is uncertain (see Whittaker [48], for a review). Nevertheless, models, such as the Social Expansion Theory developed by Carlson and Zmud [6] and the Compensatory Adaptation Theory proposed by Kock [24], have argued that the repetitive use of a communication medium to accomplish a particular task is likely to lead over time to that medium being perceived as richer than before, notably because users adapt to what they perceive as the medium's initial lack of richness [24]. Further, Rhoads [38] has suggested that planning can improve as a result of a better understanding of the advantages and limitations of computer-mediated communication, leading to users more effectively implementing virtual strategies.

More specifically, in [13], research on communication mediated through text-based collaboration found no adverse effects of a specific text-based collaboration technology, instant messaging, on team performance in a simulated command and control task, leading to the conclusion that teams restricted to text-based collaboration performed the task as well as teams who could communicate orally. Teams restricted to text-based collaboration also provided similar ratings of their workload and situational awareness to those given by teams that communicated directly (i.e., orally).

Although, then, there is evidence that perceived obstacles arising from computer-medicated communication may not cause significant difficulties, to the best of our knowledge there is no experimental evidence that the arguments will hold in the context of a microblogging service like Twitter being used as the mediating platform for human-robot interaction.

2.2. Social networks and robotics

Indeed, the literature on the use of social networks to communicate with robots or similar artificial devices is still scarce. One notable example which is similar to the approach discussed in this paper can be seen in Ma et al. [33], where a system was devised that allowed a number of robots to be given instructions via SMS (Short Message Service), The Microsoft Network (MSN) Instant Messenger application, Facebook, and an online calendar application. In this study, the robots were given single tasks to accomplish (e.g., 'vacuum the floor' or 'check if a window is closed'). Although complex tasks, they were from a set pre-defined by the designer so there was limited control over what the robot could accomplish. The system used a combination of listeners (connected to MSN or Facebook), a message processor and a task dispatcher, as well as a response element to allow a user with an Internet connected device or mobile phone to request that the robot carry out an action. Along with potential complexities that might be encountered by naïve users in using some of the applications involved [45], it is not clear whether each message could include more than one instruction to be executed by the robot. Finally, the system did not allow bi-directional communication since no available feedback was provided from the robot to the user.

Another allied example of social networks and robotics is seen in a short video from Takashi Ogura.[3] In the video, a developer has connected a small humanoid robot to an Arduino microprocessor, which is in turn connected to his computer and then Twitter. The user can be seen sending simple commands to the robot via a Twitter feed (e.g., '@Tweetnoid do motionName1 motionName2'). However, little information is provided about how the developer accomplished this task, and it appears that the system looks for pre-set keywords separated by a space and does not actually translate the language into runtime code for the robot. For instance, when the command '@Tweetnoid do hello' was sent, the system identified the keyword 'hello' and then made the robot move its arm. As with the previous example, it is also unclear whether the system allows bi-directional communication.

Though these examples have limitations, they demonstrate that Twitter can be considered more than a human-to-human social activity tool, and that it may also be used a means of communication between people and devices.

2.3. Event detection tasks

The way in which such communication – through social media retrieval and analysis – is coordinated tends to be focused on *event detection* tasks based on language processing techniques. For these purposes an *event* can be defined as something that happens at a given place and time; therefore, the presence of participant, place, and time information could determine the existence of an event in the textual message [44]. Examples of events could be physical events, such as natural activities (e.g., earthquakes, riots, etc.), or abstract events, such as feelings and sentiments. Further, users can be defined as sensors or agents and tweets as sensory values. In other words, the user functions as a sensor of the event. If the user sends a tweet about an event, then the user is returning a positive value. A tweet can therefore be considered as a sensor reading and, as also discussed in Sakaki et al. [39], this is a crucial assumption enabling application of various methods related to sensory information.

Following these assumptions, the *event detection* problem can be reduced into detecting the pre-defined object and its location estimation. To detect events, language processing techniques are usually applied to filter messages, discarding those that are irrelevant, and analyse the identified, relevant tweets that contain the required information. In order to process tweets, researchers either use off-the-shelf trained NLP tools, normally used for analysing formal text (i.e., speech tagging and statistical methods of extraction), or they 'retrain' existing techniques to handle the informal text that tends to be found in tweets. Such retraining is necessary because, in general, NLP tools are designed to process edited texts (such as news articles and documents) and therefore perform poorly when applied to what is found in the Twitter text domain, owing to the noisy, unique and often informal style of tweets. The retraining, therefore, usually requires the annotation of large datasets, which is costly.

2.4. Event generation tasks

However, little work has been done on *event generation* in a social Web media context. Within this framework, events are generated by actuators (instead of events being detected by sensors). In line with the detection assumptions, if robots are defined as actuators or agents, the event generation problem can then be reduced into the object identification and its mapping into sequence of actions, time and location estimation. Language

[3] The video can be found at: youtu.be/OcJPKpIR31w.

processing techniques are therefore applied to filter out the irrelevant messages, then identify the event and translate it into a sequence of actions to be executed. In this framework, the selected tweets carry the required information, however events are no longer pre-defined and the language processing phase requires some knowledge representation models to generate the correct event. Moreover, care must be taken in understanding how events are 'acquired' from the text messages (see, for example, Ganier [14]).

As discussed above, the quantity and the nature of the content generated by microblogs such as Twitter, makes the distillation of information using NLP a very difficult task. Frequently, the message is a single sentence or less; the grammar used is generally informal and unstructured, relative to the pertinent domain; and the tone is conversational, and the message frequently unedited, meaning that abbreviations and errors are common.

Moreover, the message is *semi-structured* by traditional NLP definitions, since it contains some meta-data (timestamps, location, author, etc.) in addition to free-text. In semi-structured extraction, the text to be analysed is often not comprised of full sentences, but rather short phrases, and rarely includes the sort of linguistic anaphora the supports the development of meaning. Therefore, extraction from these semi-structured texts is essentially reduced to a tagging task, based on specific tokens and token features. It may also include richer representations of context, such as part-of-speech tagging, name tagging and chunking (see, for example, Foster et al. [12]), but their implementation is problematic for event detection purposes owing to the characteristics of tweets that have already been noted. Coping with repeated, changed and dropped letters, unexpected punctuation, syntactic and semantic ellipsis are among the most common problems encountered in analysing semi-structured messages [9]. Although some of these aspects are perhaps more problematic in the context of event detection, it is less clear how they 'play out' in the event generation context on which the work in this paper focuses. Indeed, the nature of event generative messages may be different from that of event driven messages in the context of controlling robots, in which this paper is interested.

That is, event generation tasks are less likely to be affected by some of the previously discussed problems associated with NLP tools that have been applied to Twitter messages up to this point. Indeed, the linguistic style used to control the robot may be less likely to deviate much from easily-understood natural language. For example, urban slang, which is often used in human-to-human tweets, is unlikely to be perceived as helpful in instructing a robot and is therefore less likely to be used. Moreover, the constraint of 140 characters placed on a tweet results in very limited syntactic and semantic contexts, which may in turn facilitate knowledge representation; this is contrary to the case for event detection, where the limited context may make the classification task harder (see, for example, Corvey et al. [8]).

An added simplification in many event generation cases is that each message corresponds to a single instance, since the granularity of the chunks tends to be compatible with the message size limitations (see, for example, Tenbrink and Winter [46], for a discussion of the granularity of chunks). In other words, the structure of the Twitter messages may be similar to the standard natural language in an event generation domain. So, in designing an event generation approach, traditional part-of-speech tagging and knowledge representation models would form a useful starting point of the approach. In particular, the task of converting user instructions into robot procedures seems comparable to the one of acquiring procedures from text introduced in Bovair and Kieras [3]. In our system, though, the task then requires a comprehension phase, performing parsing and some referential and semantic analysis to convert the input sentence into a propositional representation. In the next step, a translation process, based on a declarative representation of the knowledge, then acts on the propositional representation to construct a procedural representation. This representation of the procedure can be encoded into an executable robot program that the interpreter process accesses and executes to generate the desired event.

The application of techniques adopted from NLP research to these various steps is justified by the nature of the problem. Indeed, if messages are interpreted as constituting a workflow, where robot activities (such as *turn* or *take*) can be seen as ordered tasks within it, the dataflow can be associated with directions (*left* or *right*, for example) and sensors, and workflow patterns (such as *sequences, conjunctions* and *loops*) can be identified as the basic control structures. For each of these basic structures, rules can be defined to represent the correct combinations of tasks and dataflow using NLP corpus-based techniques. These rules are essentially isomorphic in content to the corresponding propositional representation that can be obtained using the comprehension process described above.

Other approaches are possible, of course; for example, Schumacher et al. [41] discuss two different methods based on information extraction to acquire cases as workflows. However, it is not clear whether information extraction techniques can handle anaphora problems better than – or even as well as – NLP. Also, as argued in Dufour-Lussier et al. [10], information extraction techniques may not be well equipped to deal with loops, conjunctions, etc.

Based on these premises, in the remainder of the paper we present the design of a framework for event generation tasks in a Twitter domain. The aim is to discuss an architecture to implement social media driven human–robot interaction, based on natural language models capable of dealing with social media retrieval and knowledge representation. As such, Twitter is used to generate events by sending commands and queries to a robot; in turn, the robot uses the same platform to tweet basic feedback, creating a sense of bi-directional communication.

3. Architecture

This section explains the architecture by working through a navigation scenario using the developed system. Taking a simple initial instruction, the user could instruct the robot to turn left. In order to complete this task, the user has to begin their tweet with a keyword/Twitter 'handle' that identifies the robot (for example @robotTweeter42) followed by the instruction to 'turn left'. The system should be able to distil the tweet, identifying the user command and finally generate the associated action of the robot turning left. In developing the architecture and system reported in this paper, a Finch robot was used to generate the actions.

The Finch robot is an educational tool that was developed to assist students learning how to write code (http://www.finchrobot.com). It has a number of basic sensors, including: light; temperature; obstacle; and accelerometers. It has control over two motors, an RGB Light Emitting Diode (LED) and a buzzer [28]. In this project, it is used to show the outcome of translated commands from the tweets by moving, taking photos and obtaining sensor readings for the user.

The process (depicted in Fig. 1) begins with a tweet being received from a user (or the robot) over Twitter; the next step is to translate the tweet into a form of code that can control the Finch robot. Once the code has been generated from the message, it is then processed and executed, which in turn controls the robot by moving it and collecting data from its sensors and camera. The results of the processed code are then summarised in a HyperText

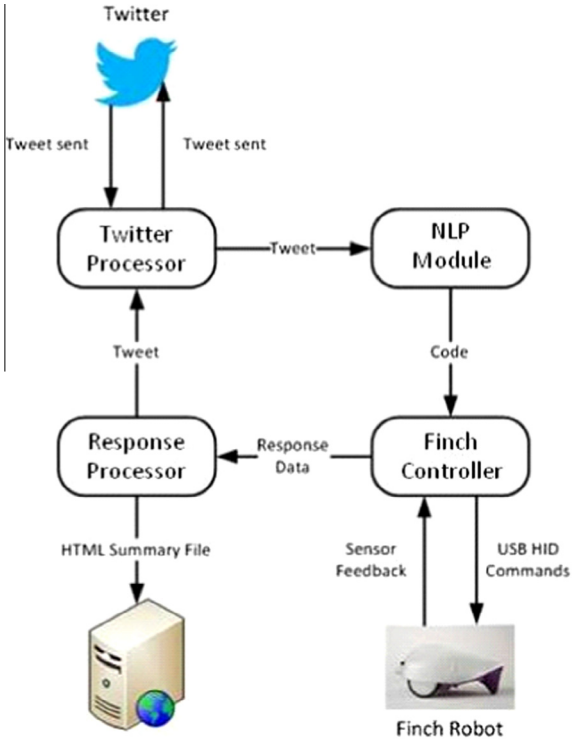

Fig. 1. High-level processes of the system.

Mark-up Language (HTML) document that shows the original tweet, the converted code and the results. This is then uploaded to a HyperText Transfer Protocol (HTTP) server, and the Uniform Resource Locator (URL) is saved in a response Tweet file. This response Tweet file is then picked up and sent back to the original user.

The architecture components for the proposed model are therefore: the *Twitter processor*; *the Natural Language Processing (NLP) module*; the *Finch controller*; and the *Response processor*. Each of these components is able to function independently of the others so that the system as a whole is more stable and is able to be modified easily without having to rebuild all of the components. To realise this architecture, the Finch controller was created using Java, whereas the Twitter processor, the NLP module and the response processor were implemented using Python. Each component will now be briefly discussed.

3.1. Twitter processor

The first component, the Twitter processor, implements a fairly simple algorithm. Once initialised, it checks for any incoming (public) tweets and then any incoming direct (private) messages (PMs). If the processor receives a message, it will store it in an eXtensible Markup Language (XML) format, as shown in Table 1. Then, it will check for any previously XML-stored messages that need to be sent to their respective users.

Once an XML-stored message has been identified, the *delivery method* is checked. If the method is a public tweet, it will send the reply to the user as a public tweet; if the method is a direct private message, it will send the response as a direct message. The message content and the sender ID are then extracted and the message is sent to the user. Once both incoming and outgoing messages have been checked and processed, the code will check for a *flag* signalling the exit from this sequence of actions. If no flag has been raised, the system will start again by checking for new messages.

Table 1
XML file format.

```
<Tweet>
<id>TWEET ID</id>
<sender>TWITTER USER</sender>
<message>MESSAGE</message>
<time>TIME OF TWEET</message>
<method>TWEET/DM</method>
</Tweet>
```

3.2. Natural language processing

The aim of this module is to translate the instructions contained in a Twitter message into a computer program. The key steps to achieve this translation – *word processing; event extraction*; *ConceptNet analysis;* and *code implementation* – are briefly discussed below.

3.2.1. Word processing

The first step in this stage of the process is to segment the message into lexical items. As a consequence, the tweet is broken down into words, or so called 'tokens'. These tokens are then used during the next step of the word processing activity: Part Of Speech (POS) tagging. During the tagging activity, each lexical item is tagged with the appropriate class.

POS is a basic form of syntactic analysis with countless applications in NLP, and has been used in attempts to develop a Twitter-tailored POS tagset and tagger (see, for example, Gimpel et al. [15]). Such tailoring attempts are motivated by the view that Twitter poses additional challenges as a result, for example, of the conversational nature of the text and the lack of conventional orthography. However, as already discussed in previous sections, it is reasonable to assume that the event generation model investigated here is closer to standard linguistic models than event detection Twitter domains and therefore an off-the-shelf Wall Street Journal (WSJ) corpus of the Penn Treebank model is used in our approach, instead of introducing more tailored POS models. The performance of a POS model based on standard corpora in the domain on which this paper focuses – navigation, which obviously contains route instructions – is one aspect that needs to be evaluated, mainly because route instructions tends to contain imperative forms which may not be frequent in standard corpora.

3.2.2. Event extraction

The second key step in the NLP process is event extraction achieved by parsing the lexical items from the word processing step. That is, once the POS tags have been assigned, commands are *chunked* by grouping consecutive words/tags together that match certain pre-defined patterns. An off-the-shelf tagger and chunker, developed by Bird et al. [2] and based on the Penn Treebank tagset (described in the previous section), was used for the grammar defined in this work, and is presented in Table 2.

Table 2 shows the grammar defined in this paper to extract a number of patterns that can group together words related to a set of basic actions to control the robot. The grammar in Table 2 has four *clauses*: ACT1, IF, LOOP, IFCLAUSE, LOOPCLAUSE. They have been designed to extract sequential, branching and repetition patterns from the user instructions contained in a tweet. The rules in Table 2 have been manually defined; 10 volunteers were invited to produce a set of route instructions so that syntactic structures could be identified. In particular, first a corpus of user instructions was collected and possible chunks identified; then, these possible

Table 2
Example of a basic set of rules for sequential constructs.

Grammar for basic *actions*
ACT1:
{<VB.*|NN|NNP><PRP>*<DT>*<NN>}
{<NNP><NN><RB><CD>}
{<NN|NNP><RB><CD><NNS>}
{<RB><VB><CD><NNS>}
{<VB.*><VB.*|RB><CD>*}
{<RB><PRP><DT><NN>}
{<NN><RB><CD>}
{<NNP><VBD>}
{<VB><NNS>}
{<NN><VBD>}
{<NNP><RB>}
{<NN><RB>}
{<NN><NNP>}
{<NNP><NNP>}
{<VBD>}
LOOP:
{<CD><NNS>}
IF:
{<IN><.*>+<NN><.*>+<JJR><.*>+<CD><.*>}
{<IN><JJ><.*>+<RBR><.*>+<CD><.*>}
IFCLAUSE:
{<IF><ACT1>}
LOOPCLAUSE:
{<ACT1><LOOP>}
{<LOOP><ACT1>}

combinations were run through the POS tagger and, finally, the relevant combinations that would represent a command were selected to produce the grammar shown in Table 2.

Each clause in Table 2 indicates how sentences should be chunked using regular expressions (or tag patterns). For example, the first clause (ACT1) has 15 rules to describe sequences of tagged words, and each of these 15 tag patterns is delimited by curly braces. In this example, the first rule says that an ACT1 chunk should be formed whenever the chunker finds any verb words (VB.*) or a noun (NN) or a Proper noun (NNP), followed by zero or more personal pronouns (PRP) followed by zero or more determiners (DT), followed by a noun (NN).

For example, the words from a tweet containing the string 'move forward' would be chunked together because the tags assigned to each term of the command would be VBD (the representation of a verb in the past tense) and RB (the representation of an adverb). This matches the rule <VB.*><VB.*|RB><CD>* to represent this user command and would therefore be classified as an action (the rule covers clauses that consist of a verb followed by another verb or adverb, and a cardinal number).

This grammar includes two recursive rules (IFCLAUSE, LOOPCLAUSE) to handle branching and loops. The patterns of a clause are executed in order. Sometimes an individual pattern will match with multiple, overlapping elements of the input. As with regular expressions and substitution more generally, the chunker will identify the first match possible, then continue looking for matches after the first match has ended. The clauses of a grammar are also executed in order.

3.2.3. ConceptNet analysis

Keyword-based and statistical approaches to Natural Language Processing (NLP) have been immensely successful, particularly for well-defined domains, but are less effective when handling the massive amounts of unstructured, human-oriented language data generated on the web. As such, there is a growing interest for solutions that allow deeper, more meaningful understanding of text. Concept-level analysis stands at the forefront of these methods for the NLP of text and social media data. This approach involves semantic analysis of text utilising wide semantic knowledge bases, such as web ontologies and semantic networks. Its application has been extremely promising in the area of sentiment and opinion analysis, which is inarguably the most complex NLP task [5].

The next step in the processing involves basic concept-level analysis that relies on ConceptNet 5 – a knowledge base that consists of common sense concepts and relations.[4] Specifically, the chunks of text output by the event extraction analysis phase (as described in the previous sub-section) are processed using ConceptNet. The knowledge in ConceptNet is drawn from a variety of sources: crowd-sourced (such as Wikipedia, Wiktionary, and the Open Mind Common Sense) and expert-created sources (such as WordNet and JMDict); and games (such as Verbosity). The aim of ConceptNet analysis is to verify the relevance of a user instruction, relying on common sense knowledge. Simply put, the system has the capacity to identify the 'chunks' that do or do not make sense (from the point of view of the robot). For example, the analysis deals with the chunk 'move forward' as the assertion: [[automobile]] is capable of [[moving forward]]. This assertion consists of two concepts, *automobile* and *moving forward*, connected by the relation, *CapableOf*. The analysis, then, determines whether the assertion is true based on its knowledge sources. In this case, the outcome is positive (an automobile can move forward); and, therefore, the chunk is verified as an activity that can be performed by the robot. Where the outcome is negative, because of the assertion not fitting with common sense knowledge, the chunk is rejected, and the instruction does not continue through the rest of the processing and execution. The robot responds by sending a tweet signalling that it did not understand the user's message. An example would be 'move upwards', which would not make sense for a robot operating in a 2D plane.

However, current knowledge sources used by ConceptNet lacking in concepts and relations that specifically suit human–robot instruction. As a result, the ConceptNet analysis implemented in the system is not currently capable of processing complex instructions that consist of if-clauses (such as 'if the temperature is lower than 25°, move forwards'). Given this limitation of the current implementation of ConceptNet analysis, the system has been configured to still feed such complex chunks to the next analysis phase which attempts to match them with pre-defined robot actions (code implementation). This phase in the analysis is detailed in the next sub-section.

3.2.4. Code implementation

The fourth and final step in the NLP module is the code implementation. This final step can be seen as an interface between the chunks produced in the previous step and the robot's pre-defined set of actions. The result produced after this final step is the program that will control the robot. That is, as a result of the event extraction step, all of the tokens relevant to a user command will have now been grouped under a chunk node. For example, if the command is a simple statement such as 'move forward', then the tokens 'move' and 'forward' will be grouped together under the chunk node "ACT1 move/VB forward/RB" as specified in Table 2. Therefore, during the final code implementation step this chunk node needs to be matched with the appropriate robot action. Table 3 shows the matching between user commands and robot actions as defined in this paper. In Table 3, the first column indicates the generic type of user action, whereas the second column shows the functional vocabulary. The functional vocabulary is the list of pre-defined procedures that can be executed by the robot. The use of a pre-defined set of robot executable actions is supported by previous research, which shows that although the number of distinct procedures increases with the number of sampled instructions, there is a decrease in new procedures identified over

[4] http://conceptnet5.media.mit.edu/.

Table 3
Range of commands implemented in the current system.

Type of command	Robot action (BNF notation)
Moving	**Move [direction] [speed]** Direction = [''forwards''|''backwards'']. Default is forwards. Speed = [''fast''|''normal''|''slow'']. Default is normal. *Example: move forward slowly*
Turning	**Turn [direction] [degrees]** Direction = [''left''|''right'']. Default is left. Degrees = any integer between 0 and 180. Default is 90. *Example: turn left 180*
Get Sensor Reading	**Get [sensor]** Sensor = [''light''|''temperature''|''obstacle''|''photo''] *Example: get temperature*
Sleeping	**Sleep [seconds]** Seconds = any integer between 0 and 100. Default is 1. *Example: sleep 20*
Conditionals	**If [sensor] [comparison] [integer] then [action]** Sensor = [''light''|''temperature''|''obstacle''|''photo''] Comparison = [''higher''|''lower''|''same''|''different''] Integer = any integer over 1 Action = [''move''|''turn''|''get''|''sleep''] *Example: If temperature is higher than 10 then turn left*
Loop	**Action integer times** Action = [''move''|''turn''|''get''|''sleep''] Integer = any integer over 1 *Example: move forward 10 times*

time. In other words, there is a tendency to rely on previously used expressions as time progresses [27].

Hence, during this final step, each chunk node is analysed. That is, within the node, the system identifies actions (VB terms such as move, turn, get, show, tell, wait) that can be matched with one of the available robot primitives from Table 3. In this way, depending on the action identified, the chunk is converted into the appropriate piece of code that is executable by the robot. During this stage, the system also searches the chunk node for the required parameters. Default behaviours are used in case the command is underspecified. For example, in the case of the chunk 'turn left', the line of code 'myFinch.turn(2,90)' would be produced and used to control the robot, where the value 2 indicates the direction (specified in the user command) and the value 90 indicates the angle of rotation, assigned by default owing to the lack of information specified in the user command. For loop and branching instructions (that is the IFCLAUSE, LOOPCLAUSE nodes, see Table 2), the same technique is applied. That is, by analysing the node, specific terms, keywords or integers are extracted to create a loop or conditional structure. Then the individual actions within the node are processed as described above to create the resulting code structure. Once all of the chunks have been processed, a final executable program is therefore produced. If no chunk nodes have been formed for a user message (for example "@twibot1 left"), then an empty program is produced.

The four analysis phases of the NLP module described above are summarised through a simple example of a user command ('move forward'), which is shown in Fig. 2.

In summary, the NLP module relies on syntactic analysis, which uses a context-free grammar that has been fine-tuned based on corpus data, and basic common sense analysis. The system also performs semantic-level analysis to produce a logical form of the sentence. For example, the instruction 'don't turn' will *not* lead to the robot turning, and the instruction 'if the temperature is higher than 25, move forward' will *not* lead to the robot moving forward, unless the condition is met. In effect, while it may appear that these phases are distinct and separate, the system implementation performs semantic interpretation in a 'rule-by-rule style', effectively interweaving the syntactic and semantic analysis processes, such that a syntactic structure is identified through being mapped into a well-formed semantic (logical form) structure.

3.3. Finch controller

Once the tweet has been translated into a piece of code, the resulting robot executable program is then passed to the Finch controller which works in a similar way to the other components of the system. The Finch controller periodically checks for any program being produced by the Translator module; if one is found, it is executed. Once the program has been processed, the results are stored (for example the user may have asked to take a picture) and the HTML generator module described below is called to produce the feedback for the user. Finally the controller will check for an exit flag and then sleep for 30 s before looking for more programs to be executed. The process is illustrated in Fig. 3.

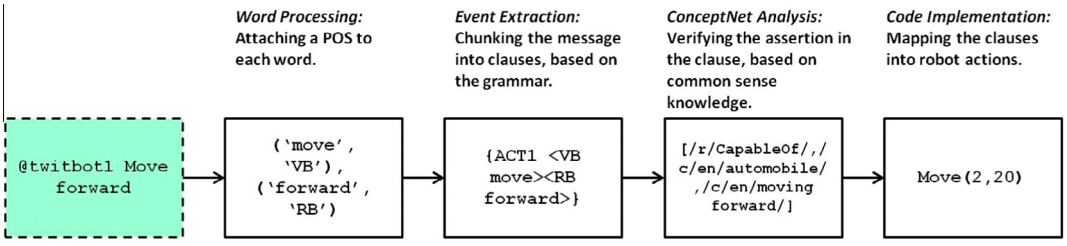

Fig. 2. The four analysis phases of the NLP module – word processing; event extraction; ConceptNet analysis; and code implementation – summarised through a tweet that contains the instruction, 'move forward'.

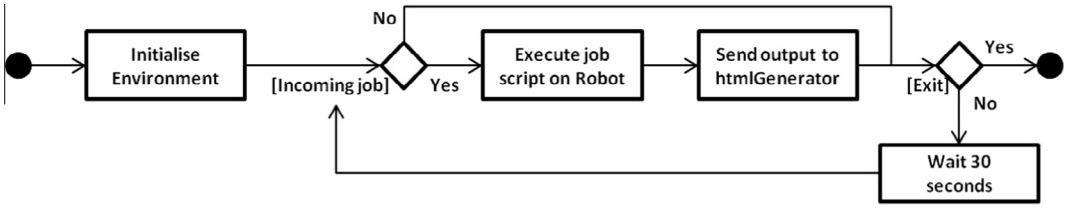

Fig. 3. Executing a procedure.

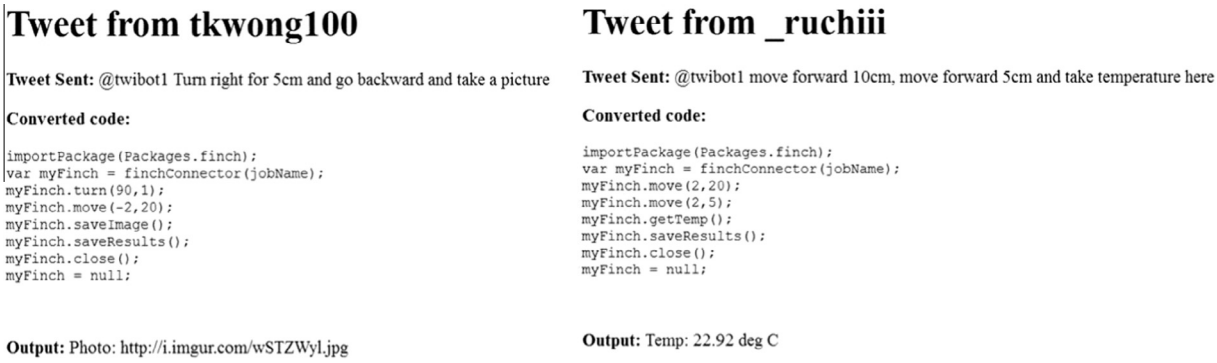

Fig. 4. Examples of HTML pages produced as a result of Tweet messages by two users in the system evaluation stage.

The JavaScript program controls the Finch robot and handles all aspects of the connection and control of the robot, including the ability to capture still images via a webcam. Table 3 shows the matching between user commands and robot actions as defined in this paper. The first column indicates the generic type of user action whereas the second column shows the *functional vocabulary*. The functional vocabulary is the list of pre-defined procedures that can be executed by the robot. A pre-defined procedure will then be implemented as robot executable program, following the process described above.

3.4. Response processor

The Response processor module creates the HyperText Mark-up Language (HTML) summary file and the response tweet that is sent back to the user. Using the Tweet ID as an argument, it collects the original Twitter message, the produced code and the output from the execution of commands in the current user commands. It then formats the collected data in a simple HTML file which is uploaded to a web server using File Transfer Protocol (FTP).

In order to ensure that the response tweet is under 140 characters long, the URL to the file is shortened using a free URL shortening service (such as that offered by tinyurl.com) and is then added to a response message. This response message is stored for the Twitter processor to then send back to the user. Fig. 4 shows examples of the HTML produced as a result of Tweet messages by two real users.

3.5. Testing

Testing focused on the ability of the system to translate Twitter messages into executable programs (event extraction phase). In particular, we tested how successful was the mapping from the sequential, branching and repetition rules defined in the grammar (see Table 2) into actions (see Table 3). For this purpose, a set of simple and different commands were tested. Table 4 provides some examples of the testing corpus. From these initial tests, it emerged that the most successful mapping occurs for messages with sequential instructions. At the same time, tweets containing loops resulted in ambiguity. For instance, for the command 'Take a photo and turn left; repeat this 4 times, then tell me the temperature' (example 5 in Table 4), the expected output should have been that the system took four photographs, turned left after each and returned the four photographs and the temperature to the user. In reality, however, the robot took a photo once, and then turned left 4 times before then returning the temperature.

Natural language is inherently and notoriously ambiguous; ambiguities are often difficult to resolve even in human-to-human interactions. For example, the last command in Table 4 could have two interpretations: a request for the robot to move forward five times and turn left five times; and a request to move forward once and turn left five times. Given that the current grammar of the system allows only one action inside a loop or conditional statement, it would correctly or incorrectly produce the second interpretation. Humans tend to resolve ambiguities through clarification requests. Human communication models describing strategies pertaining to the planning and formulation of clarification requests (see, for example, Clark [7]) have informed dialogue system development (see, for example, Skantze [43]). As such, it would be interesting to investigate how to implement such strategies in a different communication medium.

4. Usability evaluation

One of the primary aims of this research was to determine whether Twitter can be a viable platform for communication between users and interactive systems. Specific questions include: are naive users *able* to control a system using natural language; and what is their subjective *experience* of such unusual interaction situation.

Therefore, usability evaluation was a critical stage in the development of the system. According to the ISO definition [19], usability is the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". This definition postulates that in order to reliably evaluate the usability of a system, all its

Table 4
Examples of commands from the testing corpus.

User commands
Please move forward then turn left and take a photo
Go back then turn right and take a photo, finally go forwards
Tell me the temperature, then turn left and take a photo
If the temperature is lower than 25, move forwards
Take a photo and turn left; repeat this 4 times, then tell me the temperature
If the temperature is higher than 10, take a photo then move forward and turn right
Move forward then left five times, turn left then tell me the temperature and take a photo

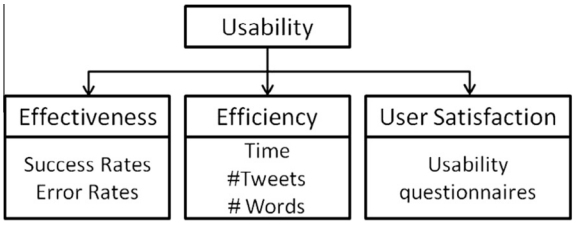

Fig. 5. Metrics used to measure each usability aspect in the study.

components, effectiveness, efficiency and satisfaction, need to be measured. Following this framework, the evaluation testing in this study relied on common effectiveness metrics, including: task completion rates and error rates (accuracy); the efficiency indicators used were task times, number of user turns and words; and, finally, user satisfaction was determined through subjective ratings and opinions collected using questionnaires. The metrics used in this study are summarised in Fig. 5. Effectiveness and efficiency measurements are relatively easy to obtain and analyse. However, measuring satisfaction is less straightforward, and the validity and objectivity of these methods are more likely to be questioned. As such, HCI practitioners largely depend on standardised questionnaires, which are shown to be more reliable than 'home-grown' ones [40], simply because they have been scientifically developed and have been repeatedly employed. The weakness of this approach, however, naturally lies in the fact that general-purpose instruments are rarely capable of probing all dimensions of a system or interface – a limitation which is exacerbated when the system is novel or experimental. Therefore, the challenge of measuring user satisfaction involves three activities: (i) identifying the questionnaire that is most suitable for the application to be evaluated; (ii) fine-tuning the questionnaire to the application; and (iii) considering complementary techniques to the questionnaire for uncovering authentic reactions and perceptions. The process adopted in this usability study is detailed in the following subsections.

4.1. Participants, task and procedure

A total of 11 participants (9 male and 2 female students at a UK university) were recruited. Participants were native or near-native speakers of English. Previous experience in using Twitter was not necessary, and no specific computer expertise or other skill was required to take part in the experiment. As seen in the pie charts in Fig. 6, participants in the study varied in terms of their familiarity with Twitter, but most of them appear to have used it before. The majority of users also opted to access Twitter using a computer.

Participants were asked to control a robot remotely using natural language commands sent via Twitter. In particular, the task was to navigate the robot to two specific locations and instruct it to take a picture and a temperature reading – these two actions could be performed in any order, and users were free to plan and modify their route as they wished. They were asked to rely on the map shown in Fig. 7, and they received written and oral instructions, but no examples or instructions were provided on how to communicate or complete the task. Participants were asked to log into Twitter, using the device of their choice (that is, Smartphone, tablet or computer). Users were informed that their interaction would be logged and written consent was obtained. After completion of the task, the users completed the 'Desirability test' and, subsequently, filled in a questionnaire. These are discussed in Section 4.3.

4.2. Efficiency and effectiveness

4.2.1. Effectiveness

Effectiveness was determined through the measures of task success and error rates. System logs indicated that all participants managed to complete the specified actions in the task (100% task success rate). The system was able to understand and translate into programmes/actions 82.5% of tweets sent (52 out of 63 tweets). This accuracy appears satisfactory, given the fact that users freely and spontaneously interacted with the system. Yet, the system failed to understand 11 tweets. Closer inspection of these instructions revealed the source of error. Seven of these utterances were out-of-grammar (for example, 'show me your position', 'go back to your starting point', 'where are you?') and the robot could not process them based on its current linguistic and functional capabilities. Users are generally inexperienced about what robots are able to say or do. In fact, it is possible that natural language interfaces encourage inflated assumptions about a system's abilities and may create expectations of human-like perception. As such, it

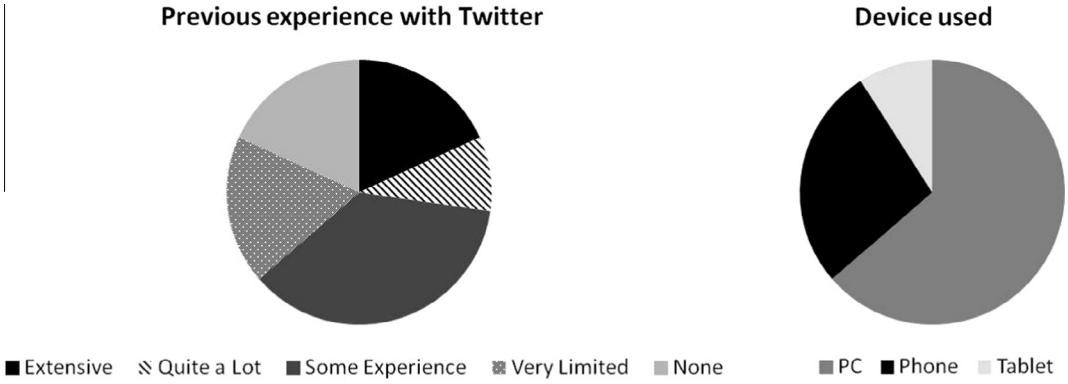

Fig. 6. Participants' previous experience with Twitter and device used to complete the study.

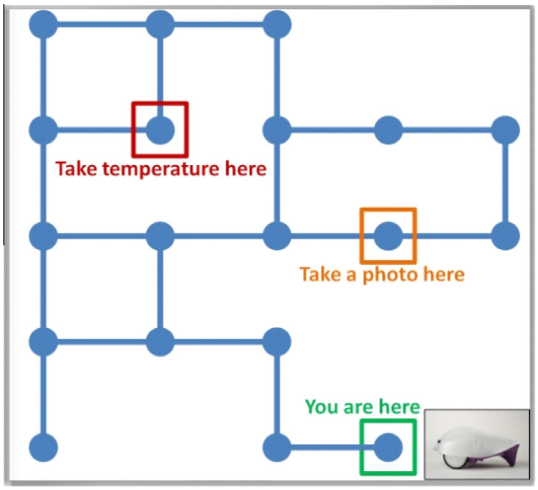

Fig. 7. The map and tasks used in the study.

appears necessary that the system is able to provide sufficient feedback to prevent or correct such misconceptions. The remaining four problematic phrases were messages such as 'I am happy' and 'I love you'. While this could simply illustrate playful behaviour, or a satisfied user, it may also relate to previous research findings that suggested that humans tend to produce social responses towards artefacts (e.g., see [35]).

4.2.2. Efficiency

Efficiency was measured using task completion times and number of turns and words required to complete the task. As shown in Table 5, the average user took approximately 17.5 min, 5.7 tweets and 52.8 words. Examination of the corpus revealed that the users of this study did not generally use any of the complex or compound utterances that were anticipated and used during system testing. In general, the vocabulary and syntactic structures of the corpus were limited and invariable (the size of the vocabulary was just 38 words). Users appear to use a strategy of reiterating expressions that had worked before. According to models of communication, people adapt their language, based on *a priori* beliefs about the addressee's knowledge. These beliefs are continuously updated based on the feedback that the addressee contributes throughout the interaction [7]. However, this process is not as straightforward in human–machine communication; users had a rudimentary initial model of the robot interlocutor's knowledge and this model was not updated in the course of the interaction, simply because the robot was unable to provide appropriate feedback. As such, the users were less willing to use different, or possibly more efficient and complex, commands (for instance, 'take the third turn right' instead of 'move forward, move forward, move forward, turn right'). A different explanation may be that people are discouraged to use complex language because of the restrictions imposed by the Twitter medium itself. Ultimately, the lack of adaptation and usage of complex language is highly desirable for a system with basic linguistic capabilities, such as the system in this study. But it is possible that users perceive such limitations as originating from the Twitter platform and not from the system itself. This idea appears to be supported by the user satisfaction data, discussed in the following sections.

Table 5
Mean time, number of tweets and words required to complete the task.

Task time (min)	Number of tweets	Number of words
17.55	5.72	52.81

4.3. User satisfaction

4.3.1. Questionnaire instrument

As argued above, it is advisable to use a standardised questionnaire for usability testing. At the same time, a degree of customisation is necessary so as to target the particular research question and domain of the study. Well-known questionnaire instruments, such as the Questionnaire for User Interaction Satisfaction (QUIS) [16], and the Software Usability Measurement Inventory (SUMI) [23], have been extensively used to evaluate systems, ranging from websites to software suites. However, systems with natural language interfaces possess different characteristics, which may not be readily addressed by these instruments. Therefore, the questionnaire selected for this study was the SASSI tool [17], which was mainly designed for and validated using speech input systems. It contained 50 seven-point Likert scale negative and positive statements (for instance, 'the system is accurate' and 'the interaction with the system is boring') which were mapped across six dimensions: System Response Accuracy, Likeability, Cognitive Demand, Annoyance, Habitability and Speed. These dimensions largely correspond to the ones found in other general-purpose questionnaire instruments, such as QUIS and SUMI. SASSI was considered a good fit for the system reported here because it is one of the few accepted tools for dialogue system evaluation. At the same time, the differences in the interaction domain and specific aims of the current study necessitated customisation of SASSI to exclude items or include additional ones. First, the 'speed' dimension addressed in SASSI was not deemed necessary at this stage of the work. Second, it was felt that SASSI did not sufficiently probe the quality of feedback provided by the system. Third, the current study aimed to go beyond assessing whether the system was perceived to be useful or user-friendly, placing greater focus on issues to do with non-task related aspects of affect and emotions (termed 'emotional usability' by Logan [32]). To this end, the SASSI questionnaire was adapted to integrate elements from other standard questionnaire instruments. In particular, the final selection of each questionnaire item for measuring a specific dimension was driven by (i) its relevance to our research objectives and (ii) its occurrence in at least one more questionnaire. The questionnaires used were: SUMI, PSSUQ [29] and SUS [4]. For items targeting 'affect' and 'system feedback', the study drew on models such as the ones by Jordan [21], Logan [32] and Kwahk and Han [26] – in which pleasure to use a product is a prominent concept – and questionnaires such as QUIS and PUTQ (Purdue Usability Testing Questionnaire) [30]. Table 6 shows the questionnaire employed in this study. The second column indicates the questionnaire instrument(s) from which each item originates. The first column shows the SASSI dimension addressed by the questions.

As can be seen in Fig. 8, the results of the questionnaire analysis indicate an overall satisfaction with the system. The high affect scores suggested that users enjoyed using the system. This is reiterated by the low scores in the Annoyance items. Indeed, it is increasingly recognised that, as technology becomes pervasive, interactive products not only need to be useful and usable, but also fascinating and appealing to use [32]. In addition, users perceived the interaction to be effortless and comfortable. Moreover, the system was perceived to be accurate. At the same time, the system received comparatively lower ratings for the habitability measure. Habitability, here, overlaps with the concept of visibility [36]. Users in this study appear to have been unable to determine what to say to the system and what the system was doing. This may be attributed to the lack of previous familiarity with similar NLP applications, but, most likely, to the absence of visual feedback and effective verbal feedback by the system.

Table 6
The items of the questionnaire completed by users at the end of the session.

Dimension	Questionnaire Item	Source of Item
Accuracy	The system didn't always do what I wanted	SASSI, SUMI
	The system didn't always do what I expected	SASSI, SUMI
	The interaction with the system is consistent	SASSI, SUMI, SUS
	The interaction with the system is efficient	SASSI, PSSUQ
Likeability/Affect	The system is pleasant	SASSI, PSSUQ, Kwahk and Han [26]
	I was able to recover easily from errors	SASSI, SUS
	I enjoyed using the system	SASSI, SUMI, PSSUQ
	It is easy to learn to use the system	SASSI, SUMI, PSSUQ, SUS
	I would use this system again	SASSI, SUS
	I felt in control of the interaction with the system	SASSI, SUMI
	I felt excited when using the system	Jordan [21]
Cognitive Demand	I felt confident and comfortable using the system	SASSI, SUS
	I felt tense using the system	SASSI, SUMI, PSSUQ
	The system is easy to use	SASSI, SUS, PSSUQ
Habitability	The interaction with the system is natural	SASSI
	I sometimes wondered if I was using the right word	SASSI, SUMI
	The feedback from the system is clear	PUTQ, QUIS, SUMI, PSSUQ, Kwahk and Han [26]
	I was not always sure what the system was doing	SASSI
Annoyance	The interaction with the system is boring	SASSI, SUMI
	The interaction with the system is frustrating	SASSI, SUMI

4.3.2. Desirability test

As mentioned in the previous section, usability testing should also aim to determine whether an application would be desirable by users. Questionnaires may not specifically target aspects of 'desirability' and 'joy of use'. Even if they do (for instance, SUMI and SASSI), they are practically limited by the keyword on which they are anchored ('this system is *pleasant*'). In response to this limitation, a method was developed and used by Microsoft R&D to collect feedback on the desirability of a new product [1]. The method consisted of a series of 118 'product reaction' words, such as 'familiar', 'overwhelming', and 'intimidating', from which the users could select the ones that most closely matched their opinions for the system they had just used. Moreover, it was argued that this approach offers participants more 'freedom' to be critical of the system [1]. As such, the study employed this technique to elicit additional insight into user perceptions. Immediately after completing the task (and before being given the questionnaire), participants were presented with the list of words in randomised order and they had to pick five words which most accurately reflected their opinions. The data obtained are presented as a 'Word Cloud' (see Fig. 9). The Word Cloud illustrates that the prevalent sentiment of participants was expressed through keywords such as, easy to use (selected by 73% – 8 out of 11 – users), creative, accessible, clear (selected by 36% of users), fun, useful, and comfortable (selected by 27% of users). These findings appear to echo the results of the questionnaire analysis, presented in the previous section. It should be noted that in Benedek and Miner [1], researchers also engaged in a discussion with users regarding the selection of each keyword. This was not performed in the current study, due to practical limitations.

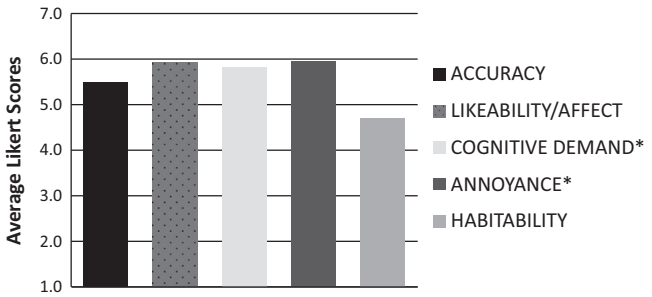

Fig. 8. Average scores for each construct. The scores for Cognitive Demand and Annoyance are reversed to facilitate comparison with the rest dimensions. The scores shown above for these dimensions were calculated by subtracting their actual average score from 7.

4.3.3. Analysis of user statements

After completing the desirability test, participants were asked to describe three positive and three negative aspects of their interaction with the system. While participants' responses were quite brief, they harmonise with the results from both questionnaires. Table 7 summarises the recurring themes (mentioned by three or more participants) and their occurrence frequency, and provides examples of statements.

As expected, the most prevalent positive perception was that the system was easy to use. Participants positively commented on the system's language understanding capabilities. This is an interesting observation, considering that the actual capabilities of the system were rudimentary. Participants also expressed that they enjoyed using the system. Most importantly, relevant to the central research question of this study, a number of participants stated that they enjoyed using Twitter as a platform to communicate with the robot. Grounded analysis of the statements regarding negative aspects was less revealing. In particular, system response times were considered problematic by almost all participants. This is certainly true, as Twitter only allows retrieval of tweets every 30 s. Delays were possibly exacerbated by the fact that all participants were interacting with the system at the same time. Along the same lines, users commented that the responses and feedback eventually sent by the system were not sufficient or appropriate. Indeed, the HTML-based output of the current implementation seems rather awkward for a normal user, but, in a future version of the system this can be easily rectified to display action executions and the relevant links in a more intuitive way. It also emerged that the system should be able to employ advanced interactive mechanisms. As noted by Koulouri et al. [25], lack of visual co-presence (supervision) increases the need for information-rich and timely feedback by the system.

5. Conclusions and future work

This paper has presented a novel approach to social data analysis, using simple grammar-based techniques to extract features of

Fig. 9. Word Cloud showing the most frequently selected items. The colour contrast and font size of each item in the Word Cloud is derived by the frequency in which the adjective was selected. As such, larger and darker words are more popular than the lighter and smaller adjectives.

Table 7
Recurring themes in user statements.

	Theme	Frequency (%)	Example statements
Positive	Ease of use	81.8	'Very simple to use', 'It was easy to understand what commands to use and how to send them'
	Language understanding	36.4	'It does understand simple English outside of programming language', 'It understands everything!'
	Platform	27.3	'It uses a social networking platform', 'great using it with Twitter'
Negative	Response times	91.0	'The system does not reply sometimes', 'It takes time to let you know that your commands are received'
	Feedback suitability	27.3	'You cannot know exactly where the robot is', 'the output looks too complicated for a normal user'

interest from the textual data retrieved from a microblogging platform in real-time. In addition, the analysis is performed with the purpose of producing programmes which lead to executable actions by a front-end application. An alternative approach for remotely controlling domestic robots has been proposed. While the Finch Robot is primarily used in educational settings, other more advanced service robots could be 'plugged in'; for example, iRobot produces a fully programmable vacuum cleaning robot. Moreover, the front-end application, the robot, is essentially a 'proof of concept' application, showcasing the simplicity and effectiveness of the architecture for controlling domestic activities and home automation through the use of Web 2.0 sites. Of immediate relevance are applications for monitoring and controlling household devices, temperature, lighting, audio, entertainment and security systems, watering plants, and feeding pets, as well as assistive technologies for the elderly and disabled. In addition, the front-end application does not necessarily have to be a physical device, like a heating system or a vacuum cleaner, but could be a database or web-based system, like a search engine, personal digital assistant or e-commerce website. Finally, as explained below, the architecture enables multiple devices of different types to be connected to the system and interfaced through a common platform (Twitter). In effect, this allows for integration of multiple appliances and systems, leading to increased convenience and efficiency.

The architecture which integrated a mobile robot and the Twitter platform was proposed, implemented and evaluated with real users. The evaluation revealed that users were able to successfully control the robot using Twitter as an interface. Most importantly, unique insight was gained into their experience. Pleasure and ease of use were among the most prevalent and recurring reactions. It was observed, however, that users employed invariable and simplistic language. This may have been a by-product of the character constraints of Twitter. Yet, this was not perceived as a limitation of the system, but rather, the system was judged to be more capable in terms of accuracy and language understanding than it actually was.

Nevertheless, the results from the evaluation reported in this paper remain preliminary. The study needs to be replicated to involve users controlling the robot using a different communication medium; for instance, an instant messaging application that imposes no turn length restrictions (similar to the setup presented in Koulouri et al. [25]) or speech, in a collocated or remote interaction setting. The comparison of the results of both studies could provide definite answers about the efficacy of Twitter as an interaction platform between humans and users. This study used a simple robotic device to showcase the rich opportunities offered by this interface approach. The research can be extended to test the approach with different types of device and application. Given the novelty of the interaction domain, there is little understanding about how users would interact with such systems and this frames a strand of future research. While the evaluation study reported in this paper produced some preliminary data, we feel that the next step would be to conduct a study in which participants can use the platform *in the wild*, for over a sustained period of time

The implications, arising from this alternative use of social media based on the principle of users being defined as actuators rather than sensors, have been identified and discussed. In particular, the linguistic repercussions have been presented and evaluated. For example, a more complex linguistic model than previously assumed is emerging about the chunking. That is, the linguistic style seems to change depending on the purpose and context in which the social media is used. However, the constraint on the length of the messages is not necessarily a limitation during the translation process, but may prove to be a 'blessing in disguise' as it forces users to utilise simpler linguistic constructs, and as such, reduce ambiguities. Yet, ambiguities inevitably arise and adaptive techniques as well dialogue-based strategies based on human–human models must be investigated further. Nevertheless, it has been possible to successfully generate events by exploiting invariants such as functional vocabulary. The event generation paradigm discussed here could be applied to a wider domain. One possibility is to use it to generate computer programs *on the fly* and with little knowledge of the programming language syntax. However, more investigation is needed to understand how scalable this approach would be.

It should be noted that the proposed architecture is extensible. That is, the module controlling the robot is not specific to the Finch robot, but is merely a queuing mechanism. This poses an interesting opportunity, where multiple robots or devices of different types can be connected to the system and queued through the

controller class. The user can then communicate with each of them using Twitter as an interface.

Finally, the event generation paradigm discussed here also bears some affinities with Complex Event Processing (CEP) models described in Tuchin et al. [47]. Indeed, they are both based on the analysis of events in (near) real-time in order to generate immediate insight and enable immediate response to the existing conditions. The event analysis is then one of pattern detection, where the definition of a pattern is based on event relations like temporal, spatial or semantic relations. These rule-based paradigms currently rely extensively on the manual definition of the relevant rules. While in this paper it has been demonstrated that it is reasonable to provide partial rule specification, providing all the required details could be a difficult task. Moreover, rules may change over time. Therefore, mechanisms for automating both the initial definition of rules and the update of rules over time, as suggested in Tuchin et al. [47], could be developed by combining CEP models with the approach presented in this paper.

As discussed above, miscommunication is a ubiquitous phenomenon of human communication and is, indeed, more prevalent in human–computer interactions. However, Martinovsky and Traum [34] suggested that through miscommunication, interlocutors gain awareness of the state and capabilities of each other. In this light, miscommunication between humans and computers is not seen as a pathological phenomenon that should be prevented, but as a key component of longer-term successful interactions. As such, the ability to initiate clarification dialogues is essential for systems with natural language interfaces. Therefore, it is interesting to investigate and implement simple clarification mechanisms, as described by communication models [7], and assess how Twitter as medium changes the patterns of interaction.

It is evident that the NLP approach described in this paper does not readily extend to different domains and contexts, given its reliance on pre-defined syntactic and semantic rules. The domain of the applications used in this study involves task-oriented and factual natural language. As such, the proposed domain is well-served by simpler analysis approaches (like the one employed in the study) that do not necessitate deep concept-level understanding. Nevertheless, interactions with intelligent personal systems, like robots, smart homes, and digital assistants, are intended to be long-term and, indeed, 'personal'. Therefore, these interactions will greatly benefit from concept-level analysis (of sentiments) which draws on common sense or user-specific knowledge bases that encompass the spatial, temporal, physical, social, and psychological aspects of everyday life [31]. This approach will permit understanding of the implicit meaning of user instructions, which is key to achieving natural and accurate NLP. In this phase of its development, the system has limited concept-level analysis capabilities. As part of future work, the operation of the component will be advanced in two directions. First, the system currently relies on the existing knowledge sources of ConceptNet, which do not provide sufficient coverage of concepts and relations relevant to the domain of human-robot navigation and instruction. Therefore, in order to fully utilise the remarkable potential of concept-level analysis for devices, such as robots capable of acting in the real-world, it is necessary to re-engineer ontology to better serve the domain. For example, one interesting aspect is that ConceptNet is fed from DBpedia (amongst other sources). Hence, a further development currently under investigation is one where robot capabilities are described in Wikipedia (maybe even for different robot types) and as such will flow from Wikipedia (Categorisation and infoboxes) to DBpedia and then ConceptNet. Second, as previously discussed, the error handling strategy of the system lacks sophistication. ConceptNet analysis captures concepts and relations; as such, in cases in which the instruction is rejected, the identified concepts and relations may be used in the formulation of intelligent clarification requests sent by the system to the user.

In conclusion, the architecture described in this paper enables event detection and generation by a single or multiple applications, and remote communication using the internet and Web 2.0 platforms. The events detected and generated can be natural language and database data, and physical sensor/actuator events. From this perspective, this study presented an early implementation and evaluation of the *Internet of Things* concept, in which anything in the world – people, physical and virtual objects – communicate and connect to each other in intelligent ways.

References

[1] J. Benedek, T. Miner, Measuring desirability: new methods for evaluating desirability in a usability lab setting, Proc. Usab. Prof. Assoc. (2002) 8–12.
[2] S. Bird, E. Loper, E. Klein, Natural Language Processing with Python O'Reilly Media Inc., 2009.
[3] S. Bovair, D.E. Kieras, Toward a model of acquiring procedures from text, Handbook Read. Res. 2 (1991) 206–229.
[4] J. Brooke, SUS – a quick and dirty usability scale, Usab. Eval. Ind. 189 (1996) 194.
[5] E. Cambria, B. White, Jumping NLP curves: a review of natural language processing research, IEEE Comput. Intell. Mag. 9 (2014) 2.
[6] J.R. Carlson, R.W. Zmud, Channel expansion theory and the experiential nature of media richness perceptions, Acad. Manage. J. 42 (2) (1999) 153–170.
[7] H.H. Clark, Using Language, Cambridge University Press, New York, US, 1996.
[8] W. Corvey, S. Vieweg, T. Rood, M. Palmer, Twitter in mass emergency: what NLP techniques can contribute, in: Proceedings of the NAACL HLT 2010, Association for Computational Linguistics, 2010, pp 23–24.
[9] K.D. Dent, S.A. Paul, Through the twitter glass: detecting questions in micro-text, in: Analyzing Microtext, 2011.
[10] V. Dufour-Lussier, F. Le Ber, J. Lieber, E. Nauer, Automatic case acquisition from texts for process-oriented case-based reasoning, Inform. Syst. (2012).
[11] J. Dunn, Twitter Statistics You May Not Knowedudemic.com/2012/12/14-twitter-statistics-you-may-not-know, 2012.
[12] J. Foster, O. Cetinoglu, J. Wagner, J. Le Roux, S. Hogan, J. Nivre, J. Van Genabith, #hardtoparse: POS tagging and parsing the twitterverse, in: Proceedings of the Workshop on Analyzing Microtext AAAI 2011, 2011, pp. 20–25.
[13] G.J. Funke, S.M. Galster, W.T. Nelson, A. Dukes, Instant messaging and team performance in a simulated command and control environment, in: General Dynamics Advanced Information Systems Dayton OH, 2006.
[14] F. Ganier, Cognitive models of processing procedural instructions, Commun. Technol. 10 (2012) 39.
[15] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, N.A. Smith, Part-of-speech tagging for twitter: annotation, features, and experiments, in: Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2010.
[16] P.D. Harper, K.L. Norman, Improving user satisfaction: the questionnaire for user interaction satisfaction version 5.5, in: The 1st Annual Mid-Atlantic Human Factors Conference, Virginia Beach, VA, 1993, pp. 224–228.
[17] K.S. Hone, R. Graham, Towards a tool for the subjective assessment of speech system interfaces (SASSI), Natural Lang. Eng. 6 (3 and 4) (2000) 287–303.
[18] Y. Hu, K. Talamadupula, S. Kambhampati, Dude, srsly?: the surprisingly formal nature of twitter's language, in: Proceedings of ICWSM, 2013.
[19] ISO 9241-11:1998. Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) – Part 11: Guidance on Usability.
[20] A. Java, X. Song, T. Finin, B. Tseng, Why we twitter: understanding microblogging usage and communities, in: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis, ACM, 2007, pp. 56–65.
[21] P. Jordan, Designing Pleasurable Products: An Introduction to The New Human Factors, CRC Press, 2000.
[22] C. Kidd, C. Breazeal, Comparison of social presence in robots and animated characters, in: Proceedings of Human–Computer Interaction (CHI), 2005.
[23] J. Kirakowski, The software usability measurement inventory: background and usage, Usab. Eval. Ind. (1996) 169–178.
[24] N. Kock, Media naturalness and compensatory encoding: the burden of electronic media obstacles is on senders, Decis. Support Syst. 44 (2007) 175–187.
[25] T. Koulouri, S. Lauria, R.D. Macredie, S. Chen, Are we there yet?: the role of gender on the effectiveness and efficiency of user-robot communication in navigational tasks, ACM Trans. Comput.–Human Interact. (TOCHI) 19 (1) (2012) 4.
[26] J. Kwahk, S.H. Han, A methodology for evaluating the usability of audiovisual consumer electronic products, Appl. Ergonom. 33 (5) (2002) 419–431.
[27] S. Lauria, G. Bugmann, T. Kyriacou, J. Bos, E. Klein, Training personal robots using natural language instruction, IEEE Intell. Syst. 16 (5) (2001) 38–45.
[28] Lauwers, T. (2010) Aligning Capabilities of Interactive Educational Tools to Learner Goals, ERIC.
[29] J.R. Lewis, IBM computer usability satisfaction questionnaires:psychometric evaluation and instructions for use, International Journal of Human-Computer Interaction 7 (1) (1995) 57–78.

[30] H.X. Lin, Y.Y. Choong, G. Salvendy, A proposed index of usability: a method for comparing the relative usability of different software systems, Behav. Inform. Technol. 16 (4–5) (1997) 267–277.
[31] H. Liu, P. Singh, ConceptNet—a practical commonsense reasoning tool-kit, BT Technol. J. 22 (4) (2004) 211–226.
[32] R.J. Logan, Behavioral and emotional usability: Thomson consumer electronics, in: Usability in Practice, Academic Press Professional Inc., 1994, pp. 59–82.
[33] X. Ma, X. Yang, S. Zhao, C. Fu, Z. Lan, Y. Pu, Robots in my contact list: using social media platforms for human-robot interaction in domestic environment, in: APCHI12, Proceedings of the 10th Asia Pacific Conference on Computer Human Interaction, ACM, New York, NY, USA, 2012, pp. 133–140.
[34] B. Martinovsky, D. Traum, The error is the clue: breakdown in human–machine interaction, in: ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems, 2003.
[35] C. Nass, Y. Moon, Machines and mindlessness: social responses to computers, J. Soc. Issues 56 (1) (2000) 81–103.
[36] D.A. Norman, The psychology of everyday things. Basic Books, 1988.
[37] S. Petrovic, M. Osborne, V. Lavrenko, Streaming first story detection with application to twitter, in: NAACL '10: Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2010.
[38] M. Rhoads, Face-to-face and computer-mediated communication: what does theory tell us and what have we learned so far?, J Plan. Lit. 25 (2) (2010) 111–122.
[39] T. Sakaki, M. Okazaki, Y. Matsuo, Earthquake shakes twitter users: real-time event detection by social sensors, in: WWW '10: Proceedings of the 19th international conference on World wide web, ACM, New York, NY, USA, 2010, pp. 851–860.
[40] J. Sauro, J.R. Lewis, Correlations among prototypical usability metrics: evidence for the construct of usability, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2009, pp. 1609–1618.
[41] M. Schumacher, M. Minor, K. Walter, R. Bergmann, Extraction of procedural knowledge from the web: a comparison of two work extraction approaches, in: Proceedings of the 21st International Conference Companion on World Wide Web, WWW '12 Companion, ACM, New York, NY, USA, 2012, pp. 739–747.
[42] H. Shi, T. Tenbrink, Telling Rolland where to go: HRI dialogues on route navigation, in: Spatial Language and Dialogue, Oxford University Press, 2009.
[43] G. Skantze, Exploring human error recovery strategies: implications for spoken dialogue systems, Speech Commun. 45 (3) (2005) 207–359.
[44] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, M. Demirbas, Short text classification in twitter to improve information filtering, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development In Information Retrieval, ACM, New York, NY, USA, 2010, pp. 841–842.
[45] C. Tagtmeier, Facebook vs. twitter. (Cover story), Comput. Libr. 30 (2010) 6–10.
[46] T. Tenbrink, S. Winter, Variable granularity in route directions, Spatial Cognit. Comput. 9 (1) (2009) 64–93.
[47] Y. Tuchin, A. Gal, S. Wasserkrug, Tuning complex event processing rules using prediction correction paradigm, in: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, 2009.
[48] S. Whittaker, Theories and methods in mediated communication, The Handbook of Discourse Processes, 2003, pp. 243–286.