DOI: 10.2298/CSIS121202065H

An evaluation of keyword, string similarity and very shallow syntactic matching for a university admissions processing infobot

Peter Hancox[1] and Nikolaos Polatidis[2]

[1] School of Computer Science, University of Birmingham
Edgbaston, Birmingham, B15 2TT, United Kingdom
pjh@cs.bham.ac.uk
[2] Department of Applied Informatics, University of Macedonia
156 Egnatia Street, 54006, Thessaloniki, Greece
npolatidis@uom.edu.gr

Abstract. "Infobots" are small-scale natural language question answering systems drawing inspiration from ELIZA-type systems. Their key distinguishing feature is the extraction of meaning from users' queries without the use of syntactic or semantic representations. Three approaches to identifying the users' intended meanings were investigated: keyword-based systems, Jaro-based string similarity algorithms and matching based on very shallow syntactic analysis. These were measured against a corpus of queries contributed by users of a WWW-hosted infobot for responding to questions about applications to MSc courses. The most effective system was Jaro with stemmed input (78.57%). It also was able to process ungrammatical input and offer scalability.

Keywords: chatbot, infobot, question-answering, Jaro string similarity, Jaro-Winkler string similarity, shallow syntactic processing.

1. Introduction

University student recruitment administration is an application where there is potential for a large volume of enquiries of a fairly routine and predictable nature from a world-wide pool of applicants. The costs of call centres (both in terms of running the centres and recruiting and retaining a knowledgeable workforce) make such ventures unattractive. On the other hand, it should be possible to implement a technological solution beyond adding over-large FAQs to web pages. The amount and breadth of information required to answer the applicants' questions would require a large number of long FAQs with quite possibly a complex net of interrelations.

Student recruitment, particularly at graduate level, is international in outlook: in UK postgraduate computing degrees, it is not unusual for international students to outnumber UK students by two to one. Communicating with international applicants brings with it all the problems of understanding versions of

English that can vary from the excellent to the less than adequate. Most applicants have a strong commitment to accessing the information that they need to be able to make a choice of what and where to study. Varying language capabilities notwithstanding, many applicants are very articulate (and so able to frame precise requests for information) and discriminating in making their choices based on that information. Although universities work hard at providing prospectus information that ranges from detail of the structure of courses to the accommodation packages and sports facilities available, many applicants still find themselves wanting to ask highly targeted questions about, for instance, module choice deadlines and visa requirements. Given the wide range of time zones that applicants come from and that 24 hour access to staff is not a possibility, the work reported here emanated from projects to implement a web-based infobot application for answering applicants' enquiries.

The aim was to provide a service where users could ask questions in English about making applications. (The system was not intended to be able to answer highly specific questions such as the reason for rejection of a particular application.) In doing this, the system would have to be able to process input in the syntactic form of questions over the relatively narrow semantic domain of applications for graduate study asked in versions of English that are not always those of a native speaker.

A number of organizations (for instance in the UK, the Alliance & Leicester bank and the O_2 mobile phone company) have made chatbots available on their websites. Chatbots have a long history and the first implementation, ELIZA [30], is particularly famous. Recent bots owe much to the ALICE software which consists of a matching engine and an XML-based language for writing rules to transform input [1].

We argue that the use of traditional chatbots for informational systems is inappropriate. Writing of the first extensive script he prepared for ELIZA which parodied "the responses of a nondirective psychotherapist", Weizenbaum [30, p. 188] stated "I chose this script because it enabled me to temporarily sidestep the problem of giving the program a database of real-world knowledge." ELIZA was a *conversational* agent: whatever happened and wherever the dialogue went, the aim was to keep the conversation going. Being a nondirective psychotherapist, information content was low, the main tactic being to use keywords from the user's input to fill slots in the system's output. In the following, ELIZA [30, p. 2] has used *boyfriend* and *depressed* to fill pre-prepared templates:

```
User:   Well, my boyfriend made me come here.
ELIZA:  Your boyfriend made you come here.
User:   He says I'm depressed much of the time.
ELIZA:  I am sorry to hear you are depressed
```

To avoid the conversation becoming too repetitive, ELIZA used two ways of introducing variety into its responses. For each keyword there could be a number of possible responses; each was used in turn to introduce some variation. Also previous topics were stored on a stack (thus keeping a record of the conversation) so that, should it be impossible to match a keyword with a template,

a previous keyword could be revisited. This had the significant effect of making it seem as if there was some larger dialogue management taking place.

The ELIZA/ALICE model is essentially conversational: the chatbot attempts to maintain a dialogue exchange above all else. The communication of information is very much a secondary objective; hence Weizenbaum's choice of a nondirective psychotherapist.

Both the Alliance & Leicester and O_2 chatbots try to communicate information about products while trying to maintain a dialogue. In particular, they use an avatar figure to represent the computer partner in the chatbot dialogue. Although it might seem attractive from a marketing point of view to present the user with a "chatbot friend" in the hope they will bond with it, many users must be sufficiently ICT-literate and the chatbots so limited that the illusion of a conversational friend is shattered. However, behind such systems, the information content is equivalent to an over-large FAQ. This paper focuses on providing a natural language interface to a set of FAQ-like topics where the number of topics is too large for a conventional WWW-based FAQ and too small for a full database natural language interface system. While a small FAQ list ranging over a very limited topic area is usually an ideal way of presenting information, a larger FAQ list ranging over a broader topic area or areas is less effective. For the information seeker, the organization of the question list may seem unfamiliar or unintuitive and the length of the list makes is difficult to locate the perhaps small piece of information. It may seem that the FAQ writer has not predicted the user's question or the information being sought is given as the part answer to several questions. For the work presented here, the user may not find their question expressed in a form they recognize, perhaps because of differing levels of competence in the language of the FAQ [25, p. 97].

More specifically, the aims of the natural language interface investigated here can be stated as:

1. *robustness* - capable of processing well-formed English or ill-formed either because the user's command of English is poor or because of ellipsis;
2. *low cost* - such a system should use relatively simple techniques to extract meaning from input and to return outputs, thus reducing the cost of implementation and maintenance;
3. *low-skilled maintenance* - it is essential that adding to and modifying the knowledge base of the application should be as simple as possible, allowing changes to be made by IT literate rather than computer science trained colleagues.

As explained above, the context of this investigation was a system for responding to natural language enquiries about applications to MSc courses. Such a system would consist of a WWW interface to a bank of 50-100 topics (i.e. too many for a manageable unhierarchically structured FAQ). Two main ways of accessing the bank of topics were chosen:

– *keywords* - keywords were manually assigned to each topic, together with a weight in the range 1...5 (where 1 was relatively insignificant and 5 extremely significant);

– *sentences* - one or more stereotypical interrogative sentences were assigned to each topic. No weights were assigned to these sentences. (These are referred to in the remainder of the paper as “stereotypical queries”.)

In both cases, it would be relatively easy for non-computer scientists to annotate the topic banks. This system is termed an “infobot” to distinguish its informational and non-conversational functionality from that of chatbots.

Experiments were designed to assess the effectiveness of a number of methods of matching queries with either sets of keywords or stereotypical queries. The latter were also used as the source for syntactically selected sets of keywords.

2. Claims

The main claim made as a result of the experiments is that:

– A Jaro-based string similarity algorithm [10] is at least as effective as the less complex keyword-based methods tested and offers better scalability.

Sub-claims are:

– Abbreviated, terse queries (e.g. “cost of courses”) and lengthy inputs have no significant effect on the performance of the best-performing matching algorithms.
– The best performing matching algorithms are robust when processing “non-native” English.
– Matching with keywords extracted using shallow syntactic techniques offers no improvement in performance.

The methodology was first to establish a corpus of queries from users. This was used as the basis for building the keyword and sentence indexes. Then, each matching method was applied to the corpus to provide a basis of comparison.

3. Preparing a Corpus

To collect a sample of inputs, a simple keyword-based infobot for delivering admissions-related information in response to natural language queries was mounted on the WWW.

This infobot was implemented in SICStus Prolog with a PrologBeans interface to the Java front-end. Users’ inputs were delivered to the Prolog application which extracted keywords or key-phrases and used these to match with keywords or key-phrases associated with “chunks” of informational text (Fig. 1). These informational texts were created after a study of a log of email enquiries received from MSc applicants in the previous of the academic year.

The system was made accessible via the WWW to applicants for MSc courses in the School of Computer Science, University of Birmingham [23] in two phases.

Informational text	Keywords
Our programmes begin on 4th October 2010. Next academic year begins on 26th September 2011.	begin beginning 'academic year' 'starting date'
The on-line application form is at: http://apply.bham.ac.uk/cp/home/loginf.	'online application'

Fig. 1. Rules and keywords from the simple chatbot

3.1. Phase 1: Initial Testing

This was a feasibility study designed to assess whether there were informational texts missing from the system or if extra keywords needed to be added to existing information texts. A subset of about 15% of current MSc applicants were contacted by email, inviting them to use the system. Taking a random sample of the set of current applicants would have been possible but unduly complex, given that the set of applicants changed dynamically as some applications were rejected and new applications were received. Rather, all applicants with surnames beginning with 'S' or 'T' were included in the subset.[3] 121 queries were submitted by members of this subset of applicants. These were analysed with two extra informational texts being added and extra keywords added to some existing informational texts. This resulted in the infobot system that was used in the second phase to produce the corpus used in the experiments described in the remainder of this paper.

3.2. Phase 2: Corpus Collection

The second phase was used to collect a reference sample of queries that might be used to evaluate later systems, to analyse the behaviour of users and to analyse the performance of this simple system. 573 applicants were invited by email to use the system (being applicants with surnames beginning with other than 'S' or 'T'). 357 queries were recorded of which 70 were repeats[4].

All inputs and responses were logged. Each input was manually annotated as one of:

- *Correct* - the input was judged to be grammatical, correctly spelled and the question appropriate to the domain.
- *Correct/spelling error* - an otherwise correct input that contains at least one spelling error.

[3] This subset of surnames was chosen because the spread of nationalities of, and languages spoken by, applicants was better than other subsets of surnames, e.g. 'A' and 'Z'.

[4] A repeat is defined as a user immediately entering an input identical to their previous input.

Examples: How long it takes to finish the *porgram*? How do I know if my online registration is *finnished*?

- *Correct/grammar error* - an otherwise correct input that contains at least one grammatical error.
  Examples: Do i require to attend an interview? Is there any part time programs?
- *Abbreviated* - an input that was too brief (usually lacking a verbal component) for keywords to be reliably identified.
  Examples: Registration? FAQ? why Birmingham?
- *Inappropriate* - the input was either judged to be grammatical, correctly spelled but the question inappropriate to the domain or the input was not English or not natural language.
  Examples: What time is it now? What is your name? Das ist ein scholarship! MumbleJumble,ISupposeThisIsATest, ????, "; OR 1=1".

3.3. Analysis of Users' Inputs

In the email inviting applicants to take part in the trial, it was explained to them that this was a system under development that needed testing. An analysis of the input shows that a substantial number of the enquiries were well-formed and relevant English questions. Some applicants chose to use abbreviated enquiries such that they might use in a general search engine. Inevitably, in the context of a test where there was no identification of individual users, some chose to enter completely irrelevant (and thus inappropriate) queries (Fig. 2).

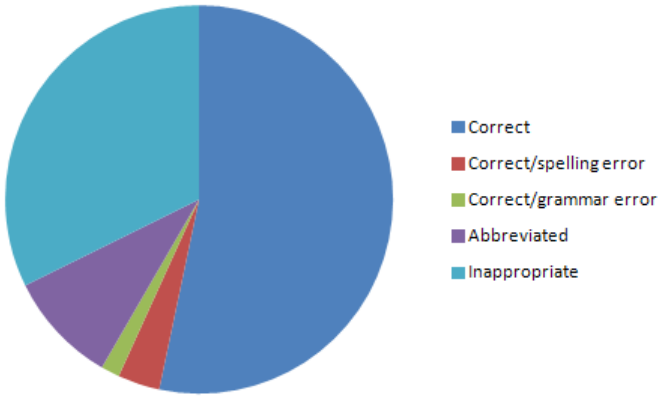

Fig. 2. Classification of inputs

From the log of inputs it could be seen that some users immediately followed their original query with one or more repetitions of the input as if they believed

Table 1. Classification of inputs

Label	Original input		Repeated input		Total input	
	n	%	n	%	n	%
Correct	105	76.64	32	23.36	137	100.00
Correct/incorrect spelling	7	77.78	2	22.22	9	100.00
Correct/incorrect grammar	4	100.00	0	0.00	4	100.00
Abbreviated	22	91.67	2	8.33	24	100.00
Inappropriate	49	59.04	34	40.96	83	100.00

that a repetition would, for some reason, return an alternative response (Table 1). It is very noticeable that users' willingness to repeat input was determined by the nature of their original input. 23.36% of correct inputs were repetitions, whereas only 8.33% of abbreviated queries were repeated, suggesting users realised that their input was too brief. The number of repetitions of inappropriate input was particularly large at 40.96%, perhaps suggesting that such users had a poor initial model of the system and were struggling to refine that model.

3.4. From Input to Corpus

To build a corpus as a tool for testing alternative designs, the inputs were selected as follows. All correct inputs were kept as were correct/grammar errors inputs. Correct inputs with spelling errors were corrected and (unless already present in the corpus) included. The inappropriate inputs were not included in the corpus. Abbreviated inputs were included where it was possible to glimpse some intended meaning. The corpus consisted of 154 queries, including well-formed and less well-formed questions as well as terse non-grammatical queries. Thus the corpus could claim to represent a real-life variety of English performance. The mean length of queries was 6.19 words and the mode was 5 words.

A "response class" set of 68 interpretations was formed. Each query in the corpus was assigned to one of the infobot's response class interpretations. For instance, the input "how long does it take to pursue a master program?" was labelled as a "duration" so that the query would be given the response "Our MSc programmes last for one year". A few response class interpretations were very closely related, for instance "birmingham_location" ("Where is Birmingham") and "location_university" ("Where is Birmingham University"). Such similarity would make the task of retrieval more difficult but reflected the practical difficulties of responding to some queries. Two topics dominated others in the corpus: the cost of tuition fees and the availability of scholarships. There was a noticeable difference between the contents of emails previously sent to admissions tutors and infobot queries: when applicants realised they were communicating with a machine, they felt sufficiently uninhibited to ask about money issues.

4. Experiments on Matching Methods

The matching methods used fell into three groups:

1. Those that used keywords extracted from the query matched against keywords assigned to interpretations from the response class (Section 4.1).
2. Those that matched the whole text of the user's query with one or more stereotypical queries assigned to interpretations from the response class (Section 4.2).
3. Shallow syntactic extraction of keywords from the user's query. The Stanford Parser [13] was used to analyse the stereotypical queries assigned to interpretations drawn from the response class, giving dictionary entries which also included information about keyword co-occurrence *and the ordering of keywords* (Section 4.3).

The results of each experiment were classified into one of three categories:

1. *Correct* - the outcome matched the expected outcome given in the corpus;
2. *Incorrect* - the outcome did not match the expected outcome given in the corpus;
3. *No response* - there was no outcome, for instance because no match was made by the current matching algorithm.[5]

4.1. Keyword-based Matching

Words judged to be significant were manually added to the keyword set.[6] In the following queries from the corpus, the keywords have been underlined:

how many modules
what is the last date of submitting the recommendations

Weights were manually assigned to each keyword, with low weight attached to meaningful but commonly used keywords ("how many" = 1) and high weight to those keywords thought to carry the main content of their queries ("recommendations" = 4). As explained above, each keyword was associated with one or more *interpretations* from the response class; an interpretation here meaning the label of a particular response, for instance the duration example (Sec. 3.4). There were 152 keywords indexing 68 topics.

Simple Keyword Matching This method of matching was not expected to be effective but was used to provide a baseline method against which all other methods could be compared. (It should be viewed as a keyword equivalent

[5] In these experiments, the use of a corpus that excluded irrelevant queries meant that "no response" would be indicative of system failure rather than irrelevant input.

[6] Here "keyword" in understood to mean both single word and multi-word keywords, e.g. "part time".

of the bag-of-words model in document classification.) In the first experiment, weights were ignored. Competing interpretations were judged solely by the number of keywords found in the input. So, if the underlined words are keywords that shared the same interpretation (*deadline_application*):

what is the last date of submitting the recommendations

the score for the *deadline_application* interpretation was 3. Where there was a tie between two or more interpretations, the first occurring interpretation was selected.[7] Results are given in Table 2.

Weighted Keyword Matching Here the weights were summed. So, if the underlined words are keywords that shared the same interpretation (*deadline_application*):

what is the last date of submitting the recommendations

and their weights were:

what - deadline_application - 1
last date - deadline_application - 3
recommendations - deadline_application - 1

the sum was 5. Where there was a tie, the first occurring interpretation was selected. Results are given in Table 3.

Simple/Weighted Keyword Matching The sum of the weights and the number of keywords found were summed. Again, using the example:

what is the last date of submitting the recommendations

where the simple keyword score was 3 and the weighted keyword score was 5, the simple weighted keyword was 8. Where there was a tie, the first occurring interpretation was selected. Results are given in Table 4. (It might seem more reasonable to calculate the mean weight of keywords by dividing the summed weight by the number of keywords but this gave a slightly worse performance.)

4.2. Sentence-based Matching (String Similarity)

One or more stereotypical queries were written for each interpretation. For instance, for the “duration” interpretation, the stereotypical queries were:

how long does a masters degree take?
how long does the program take?
how long does the programme take?

[7] In a practical system, it would be necessary to employ some principled way of choosing between tied interpretations, for instance by allowing the user to choose the response best suited to their query. This, however, is an evaluation where the emphasis is on mechanically selecting the most appropriate interpretation.

Table 2. Simple keyword matching: results

Outcome	n	%
Correct	105	68.18
Incorrect	49	31.82
No response	0	0.00
Total	154	100.00

Table 3. Weighted keyword matching: results

Outcome	n	%
Correct	118	76.62
Incorrect	36	23.38
No response	0	0.00
Total	154	100.00

Table 4. Simple and weighted keyword matching: results

Outcome	n	%
Correct	119	77.27
Incorrect	35	22.73
No response	0	0.00
Total	154	100.00

how long is the msc?
what is the duration of the course?
what is the duration of the program?
what is the duration of the programme?

The matching process was to compute the string similarity between input (here drawn from the corpus) and the stereotypical queries. There are a number of string similarity algorithms that could be used [7]. Those selected were:

- *Jaro proximity*[8] (comparing inputs/stereotypical questions forwards and backwards);
- *Jaro-Winkler proximity* (forwards and backwards).

These algorithms were devised tor comparing strings such as personal names where strings would be short and errors likely to be transpositions over fairly short distances.

The Jaro algorithm compares two strings such as ‘Martha’ and ‘Marhta’. One string is scanned, character-by-character. (In this example, ‘Martha’ is taken as the first string.) A moveable window is placed over the second string. The width of the windows is computed as half the length of the longer string - 1. The window moves in synchrony with the scanning of the first string. A match between a character in the first string can only occur within the window. In the example, the emboldened characters are matches while underlined characters are within the current window:

Martha M**a**rtha Mar**r**tha Mart**h**a Martha
Marhta M**a**rhta Ma**r**hta Marht**a** Marhta

[8] Confusingly, “proximity” and “distance” seem to be used interchangeably in the literature.

In a second scan, the number of transpositions is counted. The calculation of Jaro proximity is:

$$\frac{1}{3} \times \frac{matches}{length(string_1)} + \frac{matches}{length(string_2)} + \frac{matches - (transpositions//2)}{matches} \quad (1)$$

(It should be said that the detailed implementation of transposition matching is not intuitive: "The number of transpositions . . . is computed somewhat differently from the obvious manner." [32, p. 10].)

The Jaro-Winkler algorithm is founded on the observation that transposition errors are less likely to occur in names or addresses within the initial *n* character positions (usually *n* = 4). Winkler extended the Jaro algorithm by adding a threshold of similarity (usually 0.70). For two strings with a Jaro proximity of 0.7 or more, the initial *n* characters are matched for absolute similarity (giving a "match length"). Thus, Jaro-Winkler proximity is calculated as:

$$JaroProximity + (length(match) \times position \times (1.0 - JaroProximity)) \quad (2)$$

Jaro [10] and Jaro-Winkler [31] algorithms have a record of good performance [7]. Whilst developed for character-by-character processing of names, in these experiments the comparison was word-by-word and thus inputs in these experiments were relatively short and had a number of words comparable to the number of letters in names. The rationale was that only a very limited domain of words could be reasonably used to request information on any particular topic. Also, the form of queries could be very standardised with only minor variations, for instance because of choice of function words (e.g. "a", *v.* "the") or that there would be minor variations caused by an applicant's imperfect command of English. In both cases, a Jaro-based algorithm would seem to offer a way of pairing a stereotypical query with a closely related user query. It should be noted that the proportion of matching words (either directly aligned or transposed) was lower than the proportion of matching characters in a personal name [16].

Jaro Proximity String Similarity The standard Jaro algorithm uses a matching window defined as:

$$\frac{max(length(string_1), length(string_2))}{2} - 1 \quad (3)$$

A number of runs were tried to investigate the effect of longer window sizes, leading to the conclusion that Jaro's original window size was optimal.

Two experiments were carried out: searching from beginning to end of input/stereotypical queries (Table 5); searching from end to beginning (Table 6).

Jaro-Winkler Proximity String Similarity This modification of the Jaro algorithm rewards matches at the beginning of the two strings, specifically in the first four positions. It was used in these experiments because it seemed that the beginning of a query (e.g "how many ...", "are there any ...") was significant in the query's meaning. It was hypothesised that it would be more significant still for comparing the endings of queries because many questions in English begin with the same sequence of words, thus the endings of queries should be more discriminating. The results are presented in Tables 7 and 8.

Table 5. Jaro (forward): results

Outcome	n	%
Correct	118	76.62
Incorrect	26	23.38
No response	0	0.00
Total	154	100.00

Table 6. Jaro (backwards): results

Outcome	n	%
Correct	105	68.18
Incorrect	49	31.82
No response	0	0.00
Total	154	100.00

Table 7. Jaro-Winkler (forward): results

Outcome	n	%
Correct	117	75.97
Incorrect	37	24.03
No response	0	0.00
Total	154	100.00

Table 8. Jaro-Winkler (backwards): results

Outcome	n	%
Correct	104	67.53
Incorrect	50	32.47
No response	0	0.00
Total	154	100.00

Jaro/Jaro-Winkler with Stemming Both the forward and backwards versions of the Jaro and Jaro-Winkler algorithms were supplemented by stemming the input and stereotypical queries. The stemming algorithm used was the Porter algorithm [17]. The query:

what is the last date of submitting the recommendations

would be reduced to:

what i the last dat of submit the recommend

Results are given in Tables 9 to 12.

4.3. Sentence-based Matching (Shallow Syntax)

The hypothesis was that relative order of keywords is intrinsically important over and above mere co-occurrence of keywords. However, choice of keywords should not be left to human assignment (as in Section 4.1) but chosen using syntactic information. Additionally, the order of keywords relative to other keywords is significant as may be the distance between any two keywords.

Table 9. Jaro (forward-stemmed): results

Outcome	n	%
Correct	121	78.57
Incorrect	33	21.43
No response	0	0.00
Total	154	100.00

Table 10. Jaro (backwards-stemmed): results

Outcome	n	%
Correct	106	68.83
Incorrect	48	31.17
No response	0	0.00
Total	154	100.00

Table 11. Jaro-Winkler (forward-stemmed): results

Outcome	n	%
Correct	119	77.27
Incorrect	35	22.73
No response	0	0.00
Total	154	100.00

Table 12. Jaro-Winkler (backwards-stemmed): results

Outcome	n	%
Correct	106	68.83
Incorrect	48	31.17
No response	0	0.00
Total	154	100.00

The aim was to build a matching algorithm that, for any given keyword, had associated with it an ordered list of keywords that could (and should) occur in the query *before* the given keyword and an ordered list of keywords that could (and should) occur in the query *after* the given keyword. For the query:

what is the last date of submitting the recommendations

when the algorithm selected “submitting” as the given keyword, the ordered list of prior keywords would be [last, date] and the subsequent keywords would be [recommendations].

This method of matching required more pre-processing than the keyword-based matching and Jaro-based matching. To “learn” a set of possible keyword combinations, the Stanford Parser [13] was used to analyse each of the stereotypical queries. This produced a syntactic structure:

```
(ROOT
  (SBARQ
    (WHNP (WP what))
    (SQ (VBZ is)
      (NP
        (NP (DT the) (JJ last) (NN date))
        (PP (IN for)
          (S
            (VP (VBG submitting)
              (NP (NNS recommendations)))))))
    (. ?)))
```

from which keywords were extracted. Three sets of syntactic classes were chosen:

1. nouns and adjectives (JJ, NN, NNS)[9] giving from the example above the keywords {last, date, recommendations}.
2. verbs, nouns and adjectives (JJ, NN, NNS, VRB, VRG, VRB, VRP) giving from the example above the keywords {last, date, submitting, recommendations}.
3. WH-adverbs, verbs, nouns and adjectives (JJ, NN, NNS, VRB, VRG, VRB, VRP, WRB) giving from the example above the keywords {what, last, date, submitting, recommendations}.

For each keyword, a dictionary entry was formed giving the keyword and the ordered "before" and "after" keyword lists thus enforcing a very shallow amount of (linear) syntactic structure:

dictionary(date, [what, last], [submitting, recommendations])

The matching algorithm scanned the user's query. Each word in the input having a dictionary entry was identified as a "main keyword". The whole query was them matched as follows:

1. Each keyword in the "before" list was sought in the user's query before the occurrence of the current main keyword. If a "before" keyword was found, then any subsequent "before" keyword had to occur afterwards in the query but before the "main keyword". For instance, with the "before" keyword list [what, last], there would be a complete match with:

   what is the last date . . . ?

   but would be an incomplete match of:

   last what is the date for submitting recommendations?

   In this second example, there is an incomplete match because the algorithm requires the keywords to occur in order.

2. Each keyword in the "after" list was sought in the user's query after the occurrence of the current main keyword. The same requirement of ordering was enforced.

Keyword matches were scored. Each valid occurrence of a keyword was given a point, so

what is the last date for submitting recommendations?

scored 5, whereas:

last date is what for submitting recommendations?

scored 4 as would:

last what is the date for submitting recommendations?

[9] The Stanford Parser uses the Penn Treebank tagset.

In addition, a mean distance was calculated so that queries with fewer non-keywords between keywords would be rewarded. Given competing interpretations, the interpretation with the highest keyword score and (in the case of interpretations with the same keyword score) then with the lowest mean difference between keywords was ranked first (with the first found being arbitrarily chosen amongst equal scoring interpretations).

As stated above (page 1715), three slightly differing syntactic classes were used to construct the keyword dictionary. One of these (nouns and adjectives) was used in conjunction with the Porter stemming algorithm, so that all dictionary keywords (including those in the before and after lists) and the stereotypical queries from the test corpus were stemmed. The hypothesis was that stemming would increase the number of matching keywords and thus increase accuracy. Results for all four experiments are given in Tables 13 to 16.

Table 13. Shallow syntactic (nouns and adjectives): results

Outcome	n	%
Correct	104	67.53
Incorrect	50	32.47
No response	0	0.00
Total	154	100.00

Table 14. Shallow syntactic (verbs, nouns and adjectives): results

Outcome	n	%
Correct	105	68.18
Incorrect	49	31.82
No response	0	0.00
Total	154	100.00

Table 15. Shallow syntactic (WH-adverbs, verbs, nouns and adjectives): results

Outcome	n	%
Correct	84	54.55
Incorrect	64	41.56
No response	6	3.90
Total	154	100.00

Table 16. Shallow syntactic (nouns and adjectives) stemmed: results

Outcome	n	%
Correct	91	59.09
Incorrect	60	38.96
No response	3	1.95
Total	154	100.00

5. Interpretation of Results

The shallow syntactic matching algorithm including WH-adverbs, verbs, adjectives and nouns was the worst-performing method. The range between the worst (54.55%) and the best-performing methods (78.57%) is not particularly narrow but disappointingly poor at the high end where only four out of five queries would be correctly answered.

There is little to choose between Jaro (forward-stemmed) (78.57%), simple/weighted keywords (77.27%) and Jaro-Winkler (forward-stemmed) (77.27%).

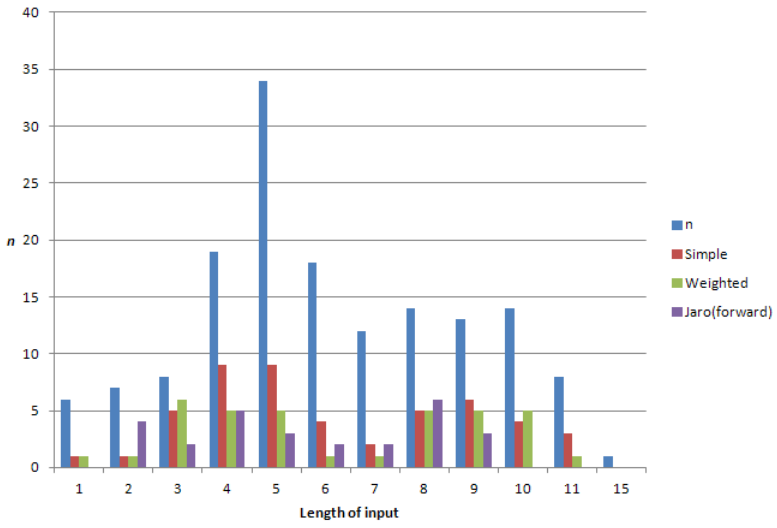

Fig. 3. Incorrect interpretations by method of matching

5.1. Simple Matching

The simple method (here used for baseline comparison) is almost the least reliable because its only strategy of choice is the number of keywords. So from Fig. 3, it can be seen that when word length rises beyond three, the simple method tends to perform less well. Essentially, given the restricted domain (and hence vocabulary of the application) the more words a query such as:

by when do I need to accept a course offer?

contains, the greater probability there is that the query contains keywords for an inappropriate interpretation. The likelihood of incorrect interpretations was compounded by the lack of a principled way of selecting between alternatives.

5.2. Simple/Weighted Matching

The performance of simple/weighted matching was almost as good as the best method. There were some queries which it processed incorrectly but which the other methods processed correctly. Examples are:

why Birmingham University?
why study at Birmingham University?

In both cases, it provided an interpretation for the very closely related query:

why should I come to Birmingham?[10]

[10] Where this sentence is understood to mean the city of Birmingham.

It seems at this level of subtlety, the simple/weighted matching method was unable to distinguish satisfactorily between competing interpretations.

5.3. Jaro (Forward-Stemmed)

Of the eight Jaro-based methods, Jaro (forward-stemmed) was most effective (78.57%). It might have been expected to work even better. Fig. 3 fails to record any particular pattern to failures amongst three similarly scoring methods, for instance they did not mainly occur amongst queries of shorter lengths. Neither did the errors occur amongst longer queries. The stemming algorithm worked to reduce variability between users' expressions. Thus users' variability of expression became less significant and one stereotypical query would match with a larger number of users' queries. This is supported by an examination of queries which keyword methods could resolve correctly but which Jaro (forward-stemmed) failed. The most extreme was:

> when do I need to finalize my course optional modules?[11]

Without a sufficiently similar stereotypical query, it would be more a matter of luck if the nearest matching stereotypical query had an appropriate interpretation.

5.4. Shallow Syntax

All four variants of the very shallow syntactic search algorithm produced consistently poor results. It would be reasonable to expect the inclusion of verbs to increase reliability as they have a crucial role in specifying complements which, in turn are realized primarily by adjectives and nouns within noun phrases. In fact, it produced no better results than the simplest (and crudest) keyword matching algorithm. The addition of stemming decreased accuracy still further. This was because it increased the number of candidate interpretations of a query without in any way contributing to an improvement in their ranking. Thus it increased the likelihood that the matching algorithm would be unable to select the current interpretation from a larger set of candidates.

5.5. Scalability

The corpus was, at 154 entries, small-scale. Nonetheless, it is possible to discern some problems of scalability even at this size. Simple/weighted matching failed where it had to choose between two closely related interpretations. An attempt to increase the success rate from 77.27% would require, in part, more keywords. This evaluation suggests that increasing keywords in a limited domain (with the likelihood that one keyword will index multiple interpretations) would bring a decrease in accuracy.

[11] A simpler way of expressing this might have been: "what is the deadline for choosing optional modules?"

While it was difficult to see a consistent pattern of failure for the best performing Jaro (forward-stemmed) (78.57%), there is some evidence that a lack of stereotypical queries was a cause of failure. Thus an increase in the coverage would, unlike simple/weighted matching, improve performance. In summary, Jaro (forward-stemmed) has the potential for scalability; simple/weighted matching does not.

5.6. Lack of Input *v.* Correctness

It was hypothesised that it would be more difficult to answer shorter queries correctly. Fig. 3 gives only very limited evidence of this. At a query length of two, Jaro (forward) did badly; at query length of three words, keyword-based matching did less well. However, there is no clearly significant evidence and so the hypothesis can be neither confirmed nor denied.

5.7. Processing Ungrammatical Inputs

It was hypothesised that simple/weighted matching would outperform Jaro (forward-stemmed) in processing ungrammatical inputs. The proportion of ungrammatical inputs[12] (less than 10%) was small. Errors of grammar (usually number/person agreement failures) were either very local ("a courses") or longer distance:

when does the university starts?

For the keyword-based systems, there was no notion of agreement: each keyword was independent and so number/person agreement could not be enforced, even if desirable. Agreement is explicit in Jaro-based methods because, assuming stereotypical queries will be well-formed, there would be no complete match between the users' inputs and the stereotypical queries. However, for longer distance ungrammaticality to be possible, there has to be a relatively long input and so the Jaro score would be less reduced than it would be with very local ungrammaticality in short inputs.

Adding stemming worked against any effects of agreement. By reducing word forms to their stems, morpho-syntactic information was removed and so it played no part in the matching process. This had the effect of improving matching.

6. Related Work

ELIZA [28] was one of three well-recognized natural language processing systems developed at much the same time. Raphael's SIR system [19] and Bobrow's STUDENT [4] answered mathematical questions. That they were both based on very formal domains of knowledge contributed to their success. As

[12] Not to be confused with abbreviated input e.g. "registration deadline?"

Weizenbaum himself noted “from the purely technical programming point of view, the psychiatric interview [which ELIZA modelled] has the advantage that it eliminates the need for storing explicit information about the real world.” [29, p. 474], [14, pp. 28-41; 110-111].

ELIZA has a long-enduring popularity. Implementations are still widely available[13] and its reputation has outlasted STUDENT and SIR. Much of ELIZA’s reputation is attributable to its domain: a psychiatric therapist was novel and still is very accessible (and even attention catching) to the less-than-expert AI practitioner. This alone does not account for its reputation and longevity: it was an early example of robust processing in that it could deal with ungrammatical input and conversations where topics were abandoned and returned to later, or the changing themes of the conversation were unrelated. Most of all, ELIZA offered an early prospect of a system that could pass the popularized notion of the Turing Test. Weizenbaum himself (incidentally rather than intentionally, it seems) raised this prospect in reporting that, although his secretary knew that in using it she was “talking to a machine”, she asked “‘Would you mind leaving the room, please?’ I [i.e. Weizenbaum] believe the anecdote testifies to the success with which the program maintains the illusion of understanding.” [29, p. 478].

Weizenbaum attempted to show the potential for question-answering systems using a refinement of the ELIZA system[14], i.e. by developing it into what this paper terms an infobot. He chose to demonstrate his ideas by providing another system to answer maths problems, which necessitated the addition of an expression evaluator. Changes were made to the store of templates. It was divided into what Weizenbaum likened to a routine (i.e. controlling set of templates which he termed a “script”) and subroutines (i.e. groups of closely related templates on very narrow topics). The reason for this hierarchical organization seems to be both practical (in that it allowed larger dialogues to be handled in small memories) and theoretical (reflecting a concern with being able to distinguish between alternative word senses). This development did not have the impact of the first ELIZA, failing to develop the ELIZA techniques in any important way, and remains a curiosity.

Shapiro and Kwasny’s 1975 development of an ELIZA infobot [24] proved more influential. In part, their success can be attributed to the development of real-time time-sharing interactive computing. Newly developing operating systems and databases were applications which allowed greater user interaction but also required quite detailed knowledge of command languages. A common theme amongst the approaches being developed to help systems was how the user could find what they needed without first knowing the appropriate technical vocabulary or command language. Shapiro and Kwasny demonstrated a straightforward development of ELIZA to provide help for the DECsystem-10.

[13] For instance: `http://www.chayden.net/eliza/Eliza.html` (Java).
[14] Confusingly Weizenbaum also called his new system ELIZA [29, p. 478, col. 1].

Their evaluation was slight by modern standards but their work was highly regarded for providing access to naïve or casual users[15] [9], [18], [20].

1991 saw the beginning of the Loebner Prize contests in which competing chatbots attempt to persuade a jury of their ability to pass the Turing Test. To succeed, it is necessary for a chatbot to chat, irrespective of topic or quality of responses. In this respect, the Loebner Prize has not directly contributed to the development of infobots. The attachment of a virtual character, an avatar, to a chatbot has become fairly common [2]. There has been a return to the chatbot as therapist with avatars being added to systems for education (e.g. [21]) and psychiatric therapy and counselling [26]. Some organizations, such as Ikea, Alliance & Leicester bank and the O_2 mobile phone company (see p. 1704 above) have used chatbots and avatar interfaces as product advisors. The development of such systems has been encouraged in part by the availability of the ALICE system with the AIML (Artificial Intelligence Markup Language) [1]. This provides a more formal way of specifying ELIZA-like templates together with a slightly more sophisticated matching algorithm which allows for some non-determinism. One addition beyond that of the original ELIZA is the "predicate" feature which allows the "botmaster" to write rules that contain is/or facts, for instance: "Samuel Clemens is Mark Twain". This provides a method, albeit unsophisticated, to store factual information.

In common with any developer of a natural language-interfaced information retrieval system, infobot developers have the twin problems of ensuring the coverage of their information resource is correct and complete and their interface covers the range of inputs users wish to employ. The capability for the user to add novel ways of expressing queries was introduced in Weizenbaum's second ELIZA [29, p. 479] where there is an impressive learning of German queries. This kind of learning was included in CSIEC, where it stored all inputs and used them in its responses [11]. The difficulty for infobots is that extending the language coverage alone might not be sufficient: it may also be necessary to extend or improve the information content. CSIEC allowed the user to add new information (i.e. by adding new facts which would be matched to existing templates and so were analysable, such as "Australia is in the Pacific"). Learning additional information by the kinds of infobot discussed in this paper would be problematic because the information to be added would usually be beyond simple facts (e.g. "Australia is in the Pacific") and the quality of newly learned information would have to be assured by the system operators.

Some infobots have been part of larger systems that include, amongst other components, a database. Sammut's system [22] provided an infobot for a museum collection. The pattern matching of natural language inputs to rules (written in production rule form) did not extend the capabilities of ELIZA (or even ALICE) and it is not clear that the incorporation of a database made any differ-

[15] Cuff [8, p. 168] offered a more rigorous analysis of what was meant by "casual user", perhaps from a less favourable standpoint when he stated: "... the author's [sic] discussion is a piece of special pleading for a natural language understanding program which will explain unfamiliar parts of a computer system."

ence to the processing of natural language inputs. Similar comments apply to a system that provides information about student loans [15]. Regretfully, there is often a lack of rigorous evaluation in this work, the notable exception being Carroll and McKendree's evaluation of three types of interface (including ELIZA) to expert systems [6].

Some researchers have, like the work reported here, tried to find alternative ways of matching inputs with FAQ-type information sources. Banchs and Li developed IRIS, a system they described as "a chat-oriented dialogue system." [3] Rather than using templates, they used a vector space model, searching over previous dialogues. Information retrieval systems for FAQs that made no claim to chatting (i.e. are less infobot and more conventional information retrieval system) have used similar, statistically-based matching techniques [12], [5]. Both of these systems identify questions within FAQs and use these to match with the users' queries. This is similar to the use of "stereotypical queries" in the current work.

Shallowness in language analysis is, it seems, a vague term. The work described here is "very shallow" in that, where is uses syntactic processing, it does so only to isolate keywords of particular grammatical classes. Wang, Ming and Chua use slightly less shallow parsing in that they (more conventionally) isolate phrase groups (e.g. VP) to find similar questions which have been asked of services such as Yahoo. They are able to claim that their technique offers robustness when presented with ungrammatical inputs [27]. At the opposite extreme, Sneiders' interpretation of shallow is, if anything, as shallow as the weighted keyword matching presented here. "Shallow language understanding" is implemented in what he terms "prioritized keyword matching" where he divides keywords into four groups: required, optional, forbidden and stop-list (i.e. high-frequency function words). There is no syntactic analysis. Perhaps to solve problems of conflicting word senses, Sneiders uses multiple lexicons, as many as one per FAQ text [25].

The work surveyed shows the tension between chatting – the need to keep the conversation going whatever the topic of ungrammaticality of the input – and the desire to provide users with information. Infobots (as opposed to chatbots) are very much rarer in the literature. While they can respond robustly to all inputs, the accuracy of their responses is disappointing. Work on developing template matching, such as reported here, has been rare. Those systems that have provided information retrieval for the kind of FAQ system used in this work have tended to use statistical techniques with limited robustness.

7. Conclusions and Further Work

A best correctness rate of 78.57% is not high enough for an effective system. The Jaro (forward-stemmed) method offers the possibility of further improvement because it is scalable, thus allowing more stereotypical queries to be used. In particular, it performed well on closely related sentences and less well on longer sentences not closely represented in the stereotypical query store.

The target application of a postgraduate application enquiry system would be used by native and non-native English speakers. There is no evidence that ungrammatical queries led to serious deterioration of performance of the Jaro (forward-stemmed) string similarity algorithm.

The problem with the use of the Jaro (forward-stemmed) method is acquiring stereotypical queries. To this end it is proposed that, in the target application, users be allowed to decide if the system has answered their question or not. If their response in positive, their query could be added to the store of stereotypical queries. Thus the system would, in a limited way, be capable of learning. In this way, it would have a more limited learning capability than those systems (e.g. CSIEC [11]) that seek knowledge from the user.

There is further work that could explore the capabilities of keyword-based searches. First, a limited dictionary of synonyms could be used to normalise queries. So, instances of “MSc”, “master”, “masters”, “MSc in”, “MSc of”, etc. could be normalised to one chosen form. This would reduce the number of keywords to be stored and make it easier to keep keywords and their weights consistent with other keywords and weights. Second, as very shallow syntactic techniques decrease effectiveness, it would seem sensible to investigate more ELIZA-like techniques by, for instance, returning to templates for matching where the system has a number of patterns of the form:

```
what is the deadline for KEYWORD(S)?
```

However, this could overcomplicate the system, leading to poorer performance. It would require a more sophisticated keyword system, perhaps of a predicate/argument structure (e.g. deadline(option_choice)). This in turn would be more difficult to use with abbreviated (“Google-like”) queries.

Acknowledgments. We wish to thank those anonymous applicants who tested the systems for us and Professor Achim Jung and Dr Alberto Simões for their support.

References

1. ALICE Artificial Intelligence Foundation: AIML: Artificial intelligence markup language, www.alicebot.org/aiml.html, [Accessed 31 May 2013].
2. Allen, C.: Artificial life, artificial agents, virtual realities: technologies of autonomous agency. In: Floridi, L. (ed.) The Cambridge Handbook of Information and Computer Ethics, chap. 13, pp. 219–233. Cambridge University Press, Cambridge (2010)
3. Banchs, R.E., Li, H.: IRIS: a chat-oriented dialogue system based on the vector space model. In: ACL 2012: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju, Jeju Island, South Korea, 8-14 July 2012. pp. 37–42. Association for Computational Linguistics, Stroudsburg, PA (2012)
4. Bobrow, D.G.: Natural language input for a computer problem solving system. In: Minsky, M.L. (ed.) Semantic Information Processing, pp. 146–226. MIT Press, Cambridge, MA (1968)
5. Burke, R.D., Hammond, K.J., Kulyukin, V., Lytinen, S.L., Tomuro, N., Schoenberg, S.: Question answering from frequently asked question files: experiences with the FAQ finder system. AI Magazine 18(2), 57–65 (1997)

6. Carroll, J.M., McKendree, J.: Interface design issues for advice-giving expert systems. Communications of the ACM 30(1), 14–32 (1987)
7. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string distance metrics for name-matching tasks. In: IIWeb-03: proceedings of the IJCAI-2003 Workshop on Information Integration on the Web, Acapulco, 9-10 August 2003. pp. 73–78. IJCAI Press, Palo Alto, CA (1993)
8. Cuff, R.N.: On casual users. International Journal of Man-Machine Studies 12(2), 163–187 (1980)
9. Houghton, R.C.: Online help systems: a conspectus. Communications of the ACM 27(2), 126–133 (1984)
10. Jaro, M.: Advances in record linkage methodology as applied to the 1985 census of tampa florida. Journal of the American Statistical Society 84(406), 414–420 (1989)
11. Jia, J.: CSIEC: a computer-assisted English learning chatbot based on textual knowledge and reasoning. Knowledge-based Systems 22(4), 249–255 (2009)
12. Jijkoun, V., de Rijke, M.: Retrieving answers from frequently asked questions pages on the web. In: CIKM '05: proceedings of the 14th conference on Information and Knowledge Management, Bremen, 31 October-5 November 2005. pp. 76–83. ACM, New York (2005)
13. Klein, D., Manning, C.: Accurate unlexicalized parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics. pp. 423–430. Association for Computational Linguistics, Stroudsburg, PA (2003)
14. Nilsson, N.J.: The quest for artificial intelligence: a history of ideas and achievement. Cambridge University Press, Cambridge (2010)
15. Owda, M., Bandar, Z., Crockett, K.: Conversation-based natural language interface to relational databases. In: WI-IATW '07: proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - workshops, Silicon Valley, CA, 5-12 Nov. 2007. pp. 363–367. IEEE Press, New York (2007), presented at Workshop on Communication between Human and Artificial Agents (CHAA-07)
16. Polatidis, N.: Chatbot for admissions. School of Computer Science, University of Birmingham (2011), unpublished MSc dissertation
17. Porter, M.: An algorithm for suffix stripping. Program 14(3), 130–137 (1990)
18. Price, L.A.: Thumb: an interactive tool for accessing and maintaining text. IEEE Transactions on Systems, Man and Cybernetics 12(2), 155–161 (1982)
19. Raphael, B.: SIR: a computer program for semantic information retrieval. In: Minsky, M.L. (ed.) Semantic Information Processing, pp. 33–145. MIT Press, Cambridge, MA (1968)
20. Relles, N., Price, L.A.: A user interface for online assistance. In: ICSE 1981: proceedings of the 5th International Conference on Software Engineering, San Diego, CA, 9-12 March 1981. pp. 400–408. IEEE Press, New York (1981)
21. Rothkrantz, L.: E-learning in virtual communities. Communication & Cognition 42(1 & 2), 35–52 (2009)
22. Sammut, C.: Managing context in a conversational agent. Linkoping Electronic Articles in Computer & Information Science 3(7) (2001)
23. Satterthwaite, S.: Prolog-Java chatbot for postgraduate admissions in the School of Computer Science. School of Computer Science, University of Birmingham (2010), unpublished MSc dissertation
24. Shapiro, S.C., Kwasny, S.C.: Interactive consulting via natural language. Communications of the ACM 8(8), 459–462 (1975)

25. Sneiders, E.: Automated FAQ answering: continued experience with shallow language understanding. In: Chaudhri, V., Fikes, R. (eds.) Question Answering Systems: papers from the 1999 AAAI Fall Symposium, North Falmouth, MA, 57 November 1999. pp. 97–107. AAAI Press, Palo Alto, CA (1999), Technical Report FS-99-02
26. Tantam, D.: The machine as psychotherapist: impersonal communication with a machine. Advances in Psychiatric Treatment 12(6), 416–426 (2006)
27. Wang, K., Ming, Z., Chua, T.S.: A syntactic tree matching approach to finding similar questions in community-based QA services. In: SIGIR '09: proceedings of the 32nd International ACM SIGIR conference on research and development in Information Retrieval Boston, MA, 19-23 July 2009. pp. 187–194. ACM Press, New York (2009)
28. Weizenbaum, J.: ELIZA: a computer program for the study of natural language communication between man and machine. Communications of the ACM 9(1), 36–45 (1966)
29. Weizenbaum, J.: Contextual understanding by computers. Communications of the ACM 10, 474–480 (1967)
30. Weizenbaum, J.: Computer power and human reason: from judgement to calculation. Penguin, Harmondsworth (1984)
31. Winkler, W.: String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In: Proceedings of the Section on Survey Research Methods. pp. 354–359. American Statistical Association, Washington, D.C. (1990)
32. Winkler, W.: Overview of record linkage and current research directions. Research report series Statistics 2006-2, Statistical Research Division, U.S. Census Bureau, Washington, D.C. (2006)

Peter Hancox is a Senior Lecturer in the School of Computer Science at the University of Birmingham. He holds a BA and was awarded a PhD for work on the machine translation of limited texts. His research interests are in both natural language processing and parallel implementations of constraint logic programming languages.

Nikolaos Polatidis is a PhD student at the Department of Applied Informatics at the University of Macedonia, Thessaloniki, Greece. He received his Master's degree in Internet Software Systems from the University of Birmingham, UK, having previously completed his BSc in Computer Science at Heriot-Watt University, Edinburgh. His research interests include recommender systems, mobile technologies and natural language processing.

Received: December 2, 2012; Accepted: August 12, 2013.