Query Matching Evaluation in an Infobot for University Admissions Processing

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 ***Formerly of:* School of Computer Science, University of Birmingham**
Edgbaston, Birmingham, B15 2TT, United Kingdom

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. Two approaches to identifying the users' intended meanings were investigated: keyword-based systems and Jaro-based string similarity algorithms. 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.

1998 ACM Subject Classification I.2.7 Natural Language Processing

Keywords and phrases chatbot, infobot, question-answering, Jaro string similarity, Jaro-Winkler string similarity

Digital Object Identifier 10.4230/OASIcs.SLATE.2012.149

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.

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.

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 [8], 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 [8, p. 188] wrote "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 [8, 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 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. While 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 NL interface system. More specifically, the aims of the NL 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 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 NL 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 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.

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 sentences. (String similarity algorithms were used for the latter.)

2 Claims

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

- A Jaro-based string similarity algorithm [3] 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.

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 pre-prepared text (Table. 1).

The system was made accessible via the WWW to applicants for MSc courses in the School of Computer Science, University of Birmingham [7] in two phases.

3.1 Phase 1: Initial Testing

This was a feasibility study in which 77 applicants (with surnames beginning with 'S' or 'T') were invited to use the system which contained templates based on email enquiries from the previous year's application round. 121 queries were submitted and outcomes analysed, allowing for the addition of further rules and, more particularly, the assignment of more keywords to texts.

Table 1 Rules and keywords from the simple chatbot.

Prepared 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'

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 to use the system (being applicants with surnames beginning with other than 'S' or 'T'). 357 queries were recorded of which 70 were repeats[1].

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.
  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! Mumble-Jumble,ISupposeThisIsATest, ????.

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. 1).

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 that a repetition

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

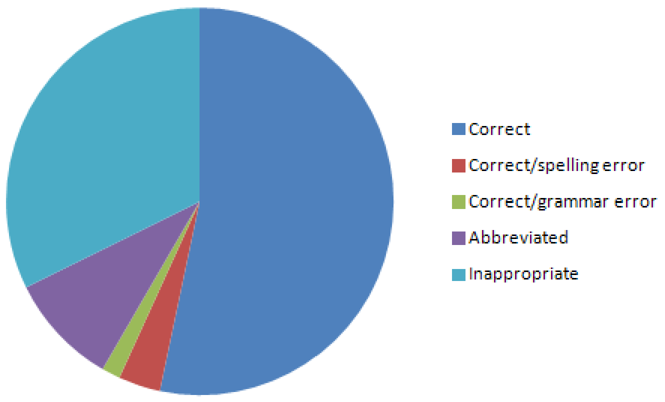

Figure 1 Classification of inputs.

Table 2 Classification of inputs.

Label	Original input n	Original input %	Repeated input n	Repeated input %	Total input n	Total input %
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

would, for some reason, return an alternative response (Table 2). 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. 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 median length of queries was 6.19 words and the mode was 5 words.

Each query in the corpus was assigned to one of the infobot's responses. 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". There were 69 distinct response classes. A few response classes 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 two groups: those that used keywords and those that used the stereotypical questions. 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* - no match was made, for instance because the user's query was irrelevant to the domain of the system.[2]

4.1 Keyword-based Matching

Words judged to be significant were manually added to the keyword set.[3] 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) to high weight to those keywords thought to carry the main content of their queries ("recommendations" = 4). Each keyword was associated with one or more *interpretations*; an interpretation here meaning the label of a particular response, for instance the duration example (Sec. 3.4). There were 152 keywords indexing 76 topics.

4.1.1 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. 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.[4] Results are given in Table 3.

[2] 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.
[3] Here "keyword" in understood to mean both single word and multi-word keywords, e.g. "part time".
[4] In a practical system, it would be necessary to employ some 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.

4.1.2 Weighted Keyword Matching

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

<u>what</u> is the <u>last date</u> of submitting the <u>recommendations</u>

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 4.

4.1.3 Simple/Weighted Keyword Matching

The sum of the weights and the number of keywords found were summed. Again, using the example:

<u>what</u> is the <u>last date</u> of submitting the <u>recommendations</u>

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 5. (It might seem more reasonable to calculate the average 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?
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 [2]. Those selected were:

- *Jaro proximity*[5] (comparing inputs/stereotypical questions forwards and backwards);
- *Jaro-Winkler proximity* (forwards and backwards).

[5] Confusingly, "proximity" and "distance" seem to be used interchangeable in the literature.

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	Ma**r**tha	Mar**t**ha	Mart**h**a	Marth**a**
$\underline{\textbf{M}\text{a}}$rhta	$\underline{\text{M}\textbf{a}\text{r}}$hta	M$\underline{\text{a}\textbf{r}\text{h}}$ta	Mar$\underline{\text{h}\textbf{t}}$a	Mar$\underline{\textbf{h}\text{ta}}$	Marh$\underline{\text{t}\textbf{a}}$

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

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

(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.” [9, 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 set to 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:

$$JaroProximity + (length(match) \times position \times (1.0 - JaroProximity)) \tag{2}$$

Jaro [3] and Jaro-Winkler [10] algorithms have a record of good performance [2]. 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. 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 [5].

4.2.1 Jaro Proximity String Similarity

The standard Jaro algorithm uses a matching window defined as:

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

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 sentences (Table 6); searching from end to beginning to end (Table 7).

4.2.2 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 8 and 9.

Table 3 Simple keyword matching: results.

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

Table 4 Weighted keyword matching: results.

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

Table 5 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

Table 6 Jaro (forward): results.

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

Table 7 Jaro (backwards): results.

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

Table 8 Jaro-Winkler (forward): results.

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

Table 9 Jaro-Winkler (backwards): results.

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

4.3 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 [6]. 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 10 to 13.

Table 10 Jaro (forward-stemmed): results.

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

Table 11 Jaro (backwards-stemmed): results.

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

Table 12 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 13 Jaro-Winkler (backwards-stemmed): results.

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

5 Interpretation of Results

Jaro-Winkler (backwards) was the worst-performing method. The range between the worst (67.53%) and the best-performing methods (78.57%) is surprisingly narrow.

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

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. 2, 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 more 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?

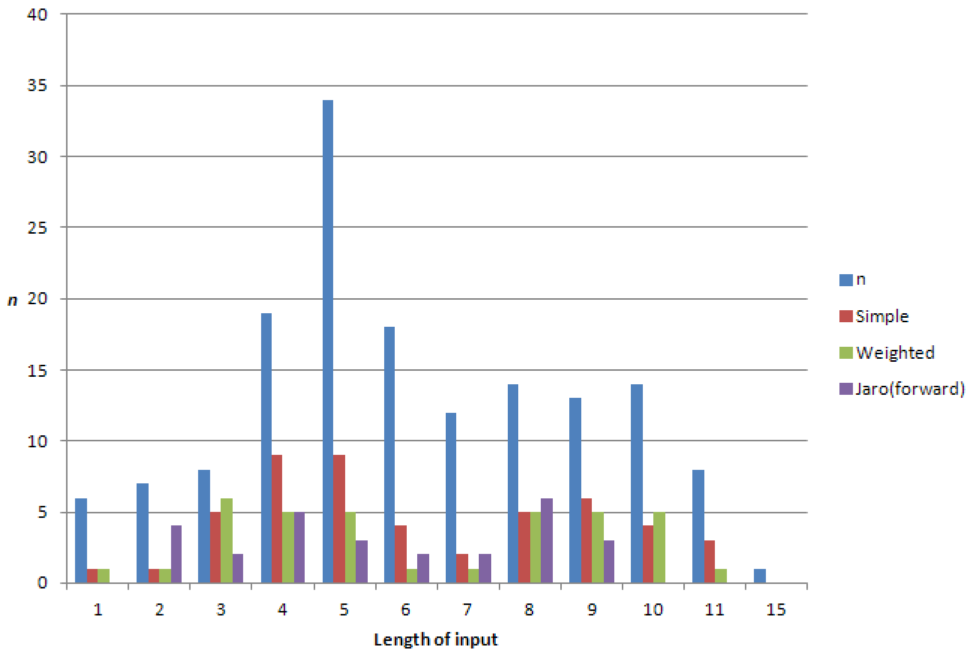

Figure 2 Incorrect interpretations by method of matching.

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

why should I come to Birmingham? [6]

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. 2 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 sentence 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? [7]

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

[6] Where this sentence is understood to mean the city of Birmingham.
[7] A simpler way of expressing this might have been: "what is the deadline for choosing optional modules?"

5.4 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.

While it was difficult to see a consistent pattern of failure for Jaro (forward-stemmed) (78.57%), there is some evidence that a lack of stereotypical sentences 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.

5.5 Lack of Input *v.* correctness

It was hypothesised that it would be more difficult to answer shorter queries correctly. Fig. 2 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.

5.6 Processing Ungrammatical Inputs

It was hypothesised that simple/weighted matching would outperform Jaro (forward-stemmed) in processing ungrammatical inputs. The proportion of ungrammatical inputs[8] (less than 10%) was small. Errors of grammar (usually number/person agreement failures) were either very local (“a courses”) or longer distance:

when <u>does</u> the university <u>starts</u>?

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 sentences will be well-formed, there would be no complete match between the users’ inputs and the stereotypical sentences. 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 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 sentences to be used. In particular, it performed well on clearly related sentences and less well on longer sentences not closely represented in the stereotypical sentence store. The target application of a postgraduate application enquiry

[8] Not to be confused with abbreviated input e.g. “registration deadline?”

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 sentences. 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 sentences. 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. Jabberwacky [4]) 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, there is a possibility of returning to ELIZA-style 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.

Acknowledgements We wish to thank those anonymous applicants who tested the systems for us.

References

1 ALICE Artificial Intelligence Foundation. *AIML: Artificial Intelligence Markup Language*. Available at: http://www.alicebot.org/aiml.html. [Accessed 9 March 2012].

2 William W. Cohen and Pradeep Ravikumar and Stephen E. Fienberg. *A comparison of string distance metrics for name-matching tasks*. In: Workshop on Information Integration on the Web (IIWeb-03). IJCAI, 2003. pp. 73–78.

3 Matthew A. Jaro. *Advances in record linkage methodology as applied to the 1985 census of Tampa Florida*. Journal of the American Statistical Society 84(406), 1989, pp. 414-–20.

4 Jiyou Jia. *CSIEC: a computer-assisted English learning chatbot based on textual knowledge and reasoning*. Knowledge-based Systems 22(4), 2009, pp. 249–255.

5 Nikolaos Polatidis. *Chatbot for admissions*. Unpublished MSc dissertation. Edgbaston: School of Computer Science, University of Birmingham, 2011.

6 M.F.Porter. *An algorithm for suffix stripping*. Program 14 (3), 1980, pp. 130-–137.

7 Rebecca Satterthwaite. *Prolog-Java chatbot for postgraduate admissions in the School of Computer Science*. Unpublished MSc dissertation. Edgbaston: School of Computer Science, University of Birmingham, 2010.

8 Joseph Weizenbaum. *Computer power and human reason: from judgement to calculation*. Penguin, Harmondsworth, 1984.

9 William E. Winkler. *Overview of record linkage and current research directions*. Washington, D.C.: Statistical Research Division, U.S. Census Bureau, 2006. (Research report series: Statistics 2006-2).

10 William E. Winkler. *String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage*. Proceedings of the Section on Survey Research Methods. American Statistical Association, Washington, D.C., 1990. pp. 354—359.