Using dialogue corpora to train a chatbot

Bayan Abu Shawar (bshawar@comp.leeds.ac.uk)
and Eric Atwell (eric@comp.leeds.ac.uk)
School of Computing, University of Leeds, Leeds LS2 9JT England

Abstract
This paper presents two chatbot systems, ALICE and Elizabeth, illustrating the dialogue knowledge representation and pattern matching techniques of each. We discuss the problems which arise when using the Dialogue Diversity Corpus to retrain a chatbot system with human dialogue examples. A Java program to convert from dialog transcript to AIML format provides a basic implementation of corpus-based chatbot training.. We conclude that dialogue researchers should adopt clearer standards for transcription and markup format in dialogue corpora to be used in training a chatbot system more effectively.
Keywords. Chatbot, matching algorithm, dialogue corpora.

1. Introduction

A chatbot is a conversational agent that interacts with users using natural language. Section two and three outline the linguistic knowledge representation and pattern matching algorithms of two chatbot systems: ALICE (ALICE 2002, Abu Shawar and Atwell 2002) and Elizabeth (Millican 2002, Abu Shawar and Atwell 2002). Both systems were adapted from the ELIZA program (Weizenbaum 1966) which emulated a psychotherapist. ALICE was found to be better suited for training using dialogue corpora because of its simple patterns templates and simple matching technique. The Dialogue Diversity Corpus (DDC) (Mann 2002) involves a collection of links to different dialogue corpuses in different domains. We used DDC samples to train ALICE, but we found several problems. Section three shows some example corpus transcripts and some problems these present. Section four presents the Java program that convert a dialogue from text to AIML format; this formalisation has helped us to see the main characteristics that must be found in dialogue corpora in order to use it for training a chatbot.

2. ALICE

ALICE (ALICE 2002, Abu Shawar and Atwell 2002): the Artificial Linguistic Internet Computer Entity, is a software robot or program that you can chat with using natural language. ALICE knowledge about English conversation patterns is stored in AIML files. AIML, or Artificial Intelligence Mark-up Language, is a derivative of Extensible Mark-up Language (XML). It was developed by the Alicebot free software community during 1995-2000 to enable people to input dialogue pattern knowledge into chatbots based on the A.L.I.C.E free software technology.

AIML consists of data objects called AIML objects, which made up of units called topics and categories. The topic is an optional top-level element, has a name attribute and a set of categories related to that topic. Categories are the basic unit of knowledge in AIML. Each category is a rule for matching an input and converting to an output, and consists of a pattern which represents the user input and a template which implies the ALICE robot answer. The AIML pattern is simple, consisting only of words, spaces, and the wildcard symbols _ and *. The words may consist of letters and numerals, but no other characters. Words are separated by a single space, and the wildcard characters function like words. The pattern language is case invariant.

3. Types of categories in ALICE

There are three types of categories: atomic categories, default categories, and recursive categories.
a. Atomic categories: are those with patterns that does not have wildcard symbols, _ and *, e.g.:
<category>
<pattern>10 DOLLARS</pattern>
<template> WOW, what a cheap.</template>
</category>
In the above category, if the user inputs: 10 dollars, then ALICE answers: WOW, what a cheap.

b. Default categories: are those with patterns having wildcard symbols * or _. The wildcard symbols match any input but they differ in their alphabetical order. Assuming the previous category, if the robot does not find the previous atomic pattern, then it will try to find the following default pattern:
<category>
<pattern>10 *</pattern>

<template>It is ten.</template>
</category>
So ALICE answers: It is ten.
c. Recursive categories: are those with templates having <srai> and <sr> tags which refers to simply recursive artificial intelligence and symbolic reduction. Recursive categories have many applications: symbolic reduction that reduces complex grammatical forms to simpler ones, divide and conquer that split an input into two or more subparts, and combine the responses to each, and dealing with synonyms by mapping different ways of saying the same thing to the same reply.

c.1 Symbolic reduction

```
<category>
<pattern>DO YOU KNOW WHAT THE * IS</pattern>
<template><srai>What is <star/></srai></template>
</category>
```

In this example <sari> is used to reduce the input to simpler form "what is".

c.2 Divide and conquer

```
<category>
<pattern>YES *</pattern>
<template><srai>YES</srai><sr/><template>
</category>
```

The input is partitioned into two parts, "yes" and the second part; * is matched with the <sr/> tag.
<sr/> = <srai><star/></srai>

c.3 Synonyms

```
<category>
<pattern>HALO</pattern>
<template><srai>Hello</srai></template>
</category>
```

The input is mapped to another form, which has the same meaning.

4. Preparation for pattern matching in ALICE
Before starting the pattern-matching algorithm, each input to the AIML interpreter must pass through two processes: normalization process, and producing input path from each sentence.
1. Normalization process involves three phases. Firstly, substitution normalization is a heuristic applied to an input that attempts to retain information in the input that would otherwise be lost during the sentence splitting or pattern fitting normalization. Secondly, sentence splitting normalization splits input into two or more sentences using rules like "break sentence at periods". Thirdly, pattern fitting normalization removes punctuation from the input, and converts input to uppercase. The normalization process is applied in this order and at least the pattern fitting normalization must be done.
Example: If the input is: "I do not know. Do you, or will you, have a robots.txt file?"

a. *Substitution Normalized form is:*
   "I do not know. Do you, or will you, have a robots dot txt file?"
b. *Sentence Splitting Normalized Form is:*
   "I do not know."
   "Do you, or will you, have a robots dot txt file"
c. *Pattern fitting Normalized Form is:*
   "I DO NOT KNOW"
   "DO YOU OR WILL YOU HAVE A ROBOTS DOT TXT FILE"

2. Producing Input Path, for each sentence an input path of the following form is produced.
Normalized-Input<that>Tvalue<topic>Pvalue. Input path has three parts. Firstly, normalized input. Secondly, <that> tag which will be followed by a Tvalue that holds the previous robot answer if it exists or * otherwise. Thirdly, <topic> tag which will be followed by Pvalue that holds the topic name if it exists or * otherwise. Topic tag is an optional top level element having a name attribute and holding all categories related to that attribute name.
Example:
If the normalized input is: "YES", previous robot output is: "DO YOU LIKE CHEESE", and there is no topic name, then the input path will be: "YES <that> DO YOU LIKE CHEESE <topic> *".

5. ALICE Pattern matching algorithm

The AIML interpreter tries to match word by word to obtain the longest pattern matching, which is the best one. This behaviour can be described in terms of the Graphmaster set of files and directories, which has a set of nodes called nodemappers and branches representing the first words of all patterns and wildcard symbols.

Assume the user input starts with word X and the root of this tree structure is a folder of the file system that contains all patterns and templates, the pattern matching algorithm uses depth first search techniques:

1. If the folder has a subfolder starts with underscore then turn to, "_/", scan through it to match all words suffixed X, if no match then:
2. Go back to folder, try to find a subfolder starts with word X, if so turn to "X/", scan for matching the tail of X, if no match then:
3. Go back to the folder, try to find a subfolder start with star notation, if so, turn to "*/", try all remaining suffixes of input following "X" to see if one match. If no match was found, change directory back to the parent of this folder, and put "X" back on the head of the input.

When a match is found, the process stops, and the template that belongs to that category is processed by the interpreter to construct the output.

Example: assume the following categories:

(1)
```
<category>
<pattern>_ WHAT IS 2 AND 2</pattern>
<template> <sr/>  <srai>WHAT IS 2 AND 2</srai> </template>
</category>
```

(2)
```
<category>
<pattern>WHAT IS 2 *</pattern>
<template> <random> <li>Two.</li> <li>Four.</li> <li>Six.</li> <li>12.</li> </random>
</template>
</category>
```

(3)
```
<category>
<pattern>HALO</pattern>
<template> <srai>HELLO</srai> </template>
</category>
```

(4)
```
<category>
<pattern>HELLO</pattern>
<template> <random> <li>Well hello there!</li> <li>Hi there!</li>
        <li>Hi there. I was just wanting to talk</li> <li>Hello there !</li> </random>
</template>
</category>
```

If the user Input is: "halo what is 2 and 2 ?", then the *ALICE output will be*: "Hi there! Four."

The process is as follows:

1- This input will match category (1) that breaks it into two sentences:
   Sentence one: represented by <sr/> tag that matches (_) which is the word HALO.
   Sentence two: WHAT IS 2 AND 2.
2- An atomic pattern is found for HALO and substituted by HELLO in category (3) and thenmatch it again with category (4); the answer will be selected randomly from the template list of category (4). Here it deals with synonyms since halo and hello has the same output.
3- The interpreter tries to match WHAT IS 2 AND 2, no atomic pattern is found, so a backtracking technique is applied, scanning to find WHAT IS 2 AND *; again no match, another backtracking tries to find WHAT IS 2 *, a match found in category (2) and a response is selected randomly from the list.
4- The AIML interpreter will combine these answers together and display them.

6. Elizabeth

Elizabeth is an adaptation of th Eliza program, in which the various selection, substitution, and phrase storage mechanisms have been enhanced and generalized to increase both flexibility and (potential) adaptability. Knowledge is stored as a script in a text file, where each line is started with a script command notation. These notations are single characters, one for each rule-type:

W: Welcome message	*Q:* quitting message	*N:* No match
V: Void input	*I:* Input transformation	*I* **:** Comment
K: Key word pattern	*R:* key word response	*M:* Memorise phrase
N: No match	*O:* Output transformation	*&:* Action to be perform

Each script command has an index code that is generated automatically. It can also be indexed using a user special code.

7. Elizabeth's Script file format

A script file may contain at most four parts. Part one, script command lines holding welcome, void and no keyword messages. Part two, input transformation rules, which starts with "I" and maps input to another form to be compatible with the defined keywords. Part three, Output transformation rules, which starts with "O" and changes personal pronouns to be appropriate as a response. Part four, keyword patterns, which starts with "K" to denote keyword pattern to be matched, followed by robot response denoted by "R". There are two type of keyword patterns: simple patterns: matching only single word, and composite patterns: match a sentence of phrase, words, string, letter, anything and nothing. For example:

```
/ The script begins with Welcome, void, no keyword and quit messages
W   HELLO, I AM ELIZABETH. WHAT DO YOU LIKE TO TALK ABOUT?
V    CAN'T YOU THINK OF ANYTHING TO SAY?
V    ARE YOU ALWAYS CHIE?
N   TELL ME WHAT YOU LIKE DOING?
Q   GOODBYE! COME BACK SOON.
/ Input transformation rules
I   MUM => mother
I   DAD => father
/Output transformation rules
O  my => YOUR
O  YOU IS => YOU ARE
/keyword transformations
K   MOTHER
K   FATHER
    R   TELL ME MORE ABOUT YOUR FAMILY.
    R   ARE YOU THE YOUNGEST IN YOUR FAMILY?
K   I LIKE [string]ING
    R   HAVE YOU [string]ED AT ALL RECENTLY?
```

If the user input is: "I like swimming", robot answer will be: "HAVE YOU SWIMMED AT ALL RECENTLY?"

8. Dynamic processing in Elizabeth

Performing a set of actions modifies the script while the conversation is under progress. These actions have the following format: "& {script command}" and you can nest as many times as you want. These actions involve adding, memorization and deleting commands. These processes can be performed directly using "!" or after matching process.

a. *Adding new script commands,* will add new script if it does not exist or replace it by new one if it is already found. Example:

```
I  my sister => my sister
  & {K  MOTHER
          & {N  DOES ANYTHING ELSE ABOUT YOUR MOTHER?}
       R  HOW WELL YOUR MOTHER AND SISTER GET ON?}
```

There are two actions to be performed. Action one, when matching occurs with the input rule, the system will search for the keyword "MOTHER", if finds then the response denoted by "R" will be added to the response list of that keyword. If not found then the outer curly brackets will be added to the script. Action two, when matching with "MOTHER" occurs later on, then a no key word message will be added to the list or replace the old one if exists.

b. *Memorization commands,* are used to hold the user or robot response for future matching. The command is denoted by M followed by memory name then the value. Example:
K [ ] MY NAME [phrase]
& {M name [phrase]}
R YOUR NAME [phrase]?
If the user input is: "my name is John" then the robot answer will be: YOUR NAME IS JOHN. And at the same time a memory name hold "John" is generated.

c. *Deleting commands* delete any script command by using the back slash notation. Example:
V\ : deletes all current void messages.
M\ one : deletes the memorization command with index "one".

9. Elizabeth's Pattern matching algorithm

Before starting the matching process, an active text is generated for each input by: converting user input to lower case, inserting spaces between words and punctuation, and removing some characters from the input, except: ! " ' ( ) , - . 0..9 : ; ? a..z

The matching process involves five stages. Stage one matches with input transformation rules. Stage two matches with keyword patterns. Stage three matches with output transformation rules. Stage four matches with void or no keyword messages. Stage five performs any dynamic process.

Stage one: input transformation rules, all input rules are applied in turn to the active text, if match occurs with the left hand side of the rule then substitute it by the right hand side, and record or apply any dynamic process found. If the same rule is applicable to the active text more than 10 times in succession, then it is applied just once.

Stage two: keyword pattern matching, all keywords are applied in turns until one match or no match at all. The response is giving according to the first match, so if there is a list of responses, one of them is selected randomly and the index code of this response will be recorded in order not to use it if the same match occurs in the next session. If there is a dynamic process, it will be applied or recorded to be performed in stage five. If no match at all found then go to phase four.

Stage three: output transformation rules, includes two steps: apply the same algorithm used in stage one. Change the active text to upper case.

Stage four: if no match occurs with the keyword patterns then either the active text is empty, and one of the void messages will be selected randomly, or no keyword found, and one of the no keyword messages will be selected.

Stage five: perform any dynamic processes, performs the recorded actions if any exists.

10. Implementing grammatical rules in Elizabeth

Elizabeth has the ability to produce a grammar structure analysis of a sentence using a set of input transformation rules to represent grammar rules. This provides an introduction to some of the major concepts and techniques of natural language processing.

Example: consider the following grammar:

S => NP VP
NP => D N
VP => V NP

This can be represented by the input rules to deal with certain nouns, verbs and determiners:

I a => (A d)
I the => (THE d)
I cat => (CAT n)
I dog => (DOG n)
I likes => (LIKES v)
I ([brak1] d) ([brak2] n) => (([brak1] D) ([brak2] N) np)
I ([brak1] v) ([brak2] np) => (([brak1] V) ([brak2] NP) vp)
I ([brak1] np) ([brak2] vp) => (([brak1] NP) ([brak2] VP) s)
K [any?]
R [any?]
W TYPE A SENTENCE USING: A, THE, CAT, DOG, RABBIT, BITES,
CHASES, LIKES

This script starts from the Welcome message, so

Robot: TYPE A SENTENCE USING: A, THE, CAT, DOG, RABBIT, BITES,

CHASES, LIKES
User input is: the cat likes the dog
Response : (((THE D) (CAT N) NP) ((LIKES V) ((THE D) (DOG N) NP) VP) s)

11. The Dialogue Diversity Corpus (DDC)

From the our analysis of ALICE and Elizabeth chatbot systems, we conclude that ALICE is more suitable for corpus-based "retraining" because AIML is closer to the markup formats used in annotated corpora and because of AIML simplicity illustrated in its pattern templates and matching algorithm. We are investigating a corpus-based approach to generalizing AIML files to cover different dialogue domains in different languages. Since natural dialogue between a person and a computer should resemble a dialogue between humans as much as possible, we have learned from dialog corpora to generate AIML files, using a program to read dialog transcripts from the Dialog Diversity Corpus (DDC) (Mann 2002) and convert these to dialogue patterns and templates in AIML format. In this section we concentrate on the problems which arise when dealing with the DDC and illustrate these problems by examples and how the program reads dialogue and converts it to AIML format.

12. Problems in the Dialog Diversity Corpus

The DDC is not a single corpus, but a collection of links to different dialogue corpuses in different fields. For example:

a. *MICAS Corpus*: Michigan Corpus of Academic Spoken English (MICASE 2002) , a collection of transcripts of academic speech events recorded at the university of Michigan. *For example:*
Astronomy transcript:
S1: circumpolar stars. So if I keep my pointer there, [S2: oh] <ROTATES CEILING>
everything else moves and we all get sick. <SS LAUGH> and we go backwards in time. And that's even more fun.
S2: make it go really really fast.
<SS LAUGH>
S1: okay so that's how the sky is going to move, a couple of other things that we can do in here, um, this is a presentation of, the, grid, that we use to divide the sky, so these lines that run, north south what do we call those?
S3: declination

The Astronomy transcript was made using a digital audio tape recorder with two microphones. Problems illustrated are: long monologs, overlapping denoted by angle brackets, more than two speakers and extra annotations recorded actions such as <SS LAUGH>.

b. *CIRCLE corpus* [6] (Center for interdisciplinary research on constructive learning environments), is a collection of transcripts holding different tutorial sessions, on topics such as physics, algebra and geometry. For example:
b.1 Algebra transcript:
2. TUTOR [Opening remarks and asks student to read out aloud and begin]
3. STUD [Reads problem] Mike starts a job at McDonald's that will pay him 5 dollars and hour, Mike gets dropped off by his parents at the start of is shift. Mike works a "h" hour shift. Write an expression for how much he makes in one night?
4. [Writes "h*5 = how much he makes"]
5. TUTOR That's right number.

b.2 Physics transcripts:
T: [student name], I'd like you to read the problem carefully, and then tell me your strategy for solving this.
S: ok
[Pause 17 sec]
hmm.
[Pause 6 sec]
T: thinking out loud as much as possible is good

The Algebra transcript was made from a video tape of algebra session where the tutor and the student are sitting in the same room. Physics transcript was done from audio form tapes of the phone conversation, where the tutor was seated in a room and students were seated in a different one. The tutor and students communicated with a conference phone that let only one person talk at a time. The

main problem here is that within the same corpora, different formats were used to distinguish speakers, and linguistic annotations such as pauses in the algebra corpus were denoted by colon whereas in the physics corpus it was directly written inside brackets.

c. CSPA Corpus of Spoken Professional American-English (Athel 2002), includes transcripts conversations of various types occurring between 1994 and 1998. *For example:*
LANGER: Hello, I'm delighted to be here.
I have carefully read and heard about the University of Albany, the State University of New York.
And I'm also the director of the National Research Center on English Learning and Achievement.
STRICKLAND: Her mother wrote the stances.
(Laughter)
KAPINUS: Dorothy, I might add also that Judith probably has more history with NAEP than just about that I know of, you know, NAEP and reading.
STRICKLAND: Yes, yes. We will really turn to you as a very important resource, Judith.
And we have a new member, Gloria Lopez Gutierrez.
And, Gloria, tell us a little bit about yourself.

In this transcript, we noticed long turns/monologues, and the transcripts were not "anonymised": speakers' last names were used.

d. The TRAINS Dialog Corpus (CISD 2002), a corpus of task-oriented spoken dialogue, that has been used in several studies of human-human dialogue. For example:
utt9 : s: okay <sli> um
utt10 : what you'll have to do is you'll have to uh pick out an <sli>
uh an engine <sli> and schedule a train to do that
utt11 : u: okay <sli> um <sli> engine <sli> two
utt12 : s: + okay +
utt13 : u: + from + Elmira
utt14 : s: + mm-hm +
utt15 : u: + to + Corning

The main problem is deling with extra-linguistic annotation like + and <sli>.

e. ICE-Singapore: International Corpus of English, Singapore English (Nelson 2002). *For example:*
<$B>
<ICE-SIN: S1A-099#33:1:B>
How how are things otherwise
<ICE-SIN:S1A-099#34:1:B>
Are you okay
<$A>
<ICE-SIN:S1A-099#35:1:A>
Uhm okay lah
<ICE-SIN:S1A-099#36:1:A>
Bearing up lah
<$B>
<ICE-SIN:S1A-099#37:1:B>
Ah hah
<$A>
<ICE-SIN:S1A-099#38:1:A>
Ya I mean I don't really feel comfortable talking about it over the phone so when I see you I'll tell you about it lah

This is a spoken dialog recorded through the phone. Problems here are: unconstrained conversations, a lot of linguistic annotation, and great variation in turn length is noticed.
<ICE-SIN:S1A-099#35:1:A> refers to text unit 35, in subtext 1, uttered by speaker A. ICE-SIN refers to corpus name ICE Singapore and S1A refers to academic spoken.

e. Mishler Book Medical Interviews (Mishler 1985) is an example of a scanned text image, including a dialogue between a patient and his physician. The problem is that scanned image was not converted to

text format and also uses extra-linguistic annotations. There are many more examples of printed books containing interview transcripts, but in general these are inaccessible if not in text-file format.

13. Desired dialogue corpus characteristics for Machine Learning

We have developed a Java program to read dialogue transcripts from the DDC and convert these to dialogue patterns and templates in AIML format in order to retrain ALICE. A lot of problems arise when dealing with the DDC, as illustrated in the previous section, which can be summarised as follows:

a. No standard formats to distinguish between speakers.
b. Extra-linguistic annotations were used.
c. No standard format in using linguistic annotations.
d. Long turns and monologues.
e. Irregular turn taking (overlapping).
f. More than one speaker.
g. Scanned text-image not converted to text format.

For example, if the program reads the dialog transcript from the astronomy transcript in MICAS corpus, every pair of speakers will generate a new AIML category where the first speaker represents the pattern, and the second speaker represents the template. But first the program calls a filtering procedure to remove the linguistic annotations such as: <SS LAUGH>, and the overlapping like: [S2: oh]. The AIML category generated is:

<category>
<pattern> CIRCUMPOLAR STARS. SO IF I KEEP MY POINTER THERE, EVERYTHING ELSE MOVES AND WE ALL GET SICK. AND WE GO BACKWARDS IN TIME. AND THAT'S EVEN MORE FUN.</pattern>
<template>make it go really really fast.</template>
</category>

It is clear that to solve these problems to obtain a good dialog model, a filtering process must be applied first to these transcripts to remove all unnecessary tags and other markup. Because of the variation in formats and annotations for theses transcripts even in the same corpus, each transcript has to be filtered and processes differently, which contradicts the generalization objective.

When re-engineering ALICE to a new domain or conversation style, the patterns and templates learnt from a training corpus are only a raw prototype: the chatbot and AIML files must be tested and revised in user trials. One of the main design considerations is how to plan the dialogue. A good dialogue design would mean less time testing and re-implementing AIML files.

In order to train a chatbot system with minimal need to post-edit the learnt AIML, dialogue corpuses should have the following characteristics:

a. Two speakers.
b. Structured format.
c. Short, obvious turns without overlapping, and without any unnecessary notes, expressions or other symbols that are not used when writing a text.

Even such "idealised" transcripts may still lead to a chatbot which does not seem entirely "natural": although we aim to mimic the natural conversation between humans, the chatbot is constrained to chatting via typing, and the way we write is different from the way we speak.

To date our machine-learnt models have not included linguistic analysis markup, such as grammatical, semantic or dialogue-act annotations (Atwell 1996, Atwell et al 2000), as ALICE/AIML makes no use of such linguistic knowledge in generating conversation responses. However, the Elizabeth chatbot does allow for more sophisticated conversation modelling including such linguistic features (see Abu Shawar and Atwell 2002).

14. Using Wmatrix to compare human and chatbot dialogues

Wmatrix (Rayson 2002) is a tool to provide a data driven method to compare between different sized corpora in three levels: word, Part-of-Speech (PoS), and semantic-tag analysis. The comparisons results are viewed as frequency lists ordered by log-likelihood ratio. LL values indicate the most important differences between corpora. We used Wmatrix to compare human-to-human dialogues extracted from the DDC corpora (Mann 2002) and human to computer dialogues extracted from chatting with ALICE (Spielberg 2000). Four different corpuses in different fields and sized were investigated against the same chatbot dialogue in semantic, PoS and lexical level, to illustrate the main

common differences between human-to-human conversation and human to computer chatting. The DDC transcripts and ALICE transcript are not equal in size, but Wmatrix compensates by providing normalised difference scores. Screenshots below illustrate the use of Wmatrix, to compare ALICE transcripts with the Astronomy human dialogue transcript extracted from the MICASE Corpus (MICASE 2002), a conversation between four speakers in an introductory astronomy discussion. The screenshots show the most important differences in semantic, Part-of-Speech and lexical levels between ALICE chatbot dialogue (file 1) and the astronomy transcript (file 2).

Sorted by log-likelihood value

Item	01	%1	02	%2	LL	
Z1	34	3.01	22	0.41 +	52.21	Personal names
E2+	16	1.42	6	0.11 +	32.44	Liking
W1	2	0.18	102	1.91 -	26.27	The universe
M6	7	0.62	151	2.82 -	24.95	Location and direction
M1	0	0.00	59	1.10 -	22.59	Moving, coming and going
H4	8	0.71	1	0.02 +	22.06	Residence
F1	6	0.53	0	0.00 +	20.97	Food
Q2.1	25	2.22	35	0.65 +	19.27	Speech etc:-
.ve						
T1.1.3	6	0.53	121	2.26 -	18.95	Time: General: Future
Q2.2	37	3.28	69	1.29 +	18.60	Speech acts

Screenshot 1; the semantic comparison

Sorted by log-likelihood value

Item	01	%1	02	%2	LL
PPIS1	55	4.88	0	0.00 +	192.23
VD0	43	3.81	27	0.50 +	67.27
PPIS2	1	0.09	129	2.41 -	41.15
CC	10	0.89	230	4.30 -	39.86
PPY	80	7.09	155	2.90 +	37.52
CS	4	0.35	116	2.17 -	23.31
ZZ1	0	0.00	56	1.05 -	21.44
DD1	9	0.80	151	2.82 -	19.97
MC	4	0.35	98	1.83 -	17.75
DD2	0	0.00	34	0.64 -	13.02

Screenshot 2; the POS comparison

Sorted by log-likelihood value					
Item	01	%1	02	%2	LL
do	44	3.90	35	0.65 +	58.69
i	54	4.79	67	1.25 +	48.04
we	1	0.09	129	2.41 -	41.15
so	1	0.09	117	2.19 -	36.75
and	8	0.71	195	3.65 -	35.19
Emily	9	0.80	0	0.00 +	31.46
you	72	6.38	151	2.82 +	28.91
this	0	0.00	70	1.31 -	26.80
you_know	8	0.71	1	0.02 +	22.06
here	0	0.00	55	1.03 -	21.06
am	6	0.53	0	0.00 +	20.97
'll	1	0.09	71	1.33 -	20.14

Screenshot 3; the word comparison

15. Conclusions

Two chatbot systems ALICE and Elizabeth were reviewed in this paper. We decide to train ALICE rather than Elizabeth to learn from human dialogue corpora for two reasons. Firstly, the AIML format is closer to the markup format used in annotated corpora. Secondly, the simplicity in generating patterns/templates, and applying simple pattern matching technique. Our main conclusion relating to Corpus Linguistics is that the Dialogue Diversity Corpus (DDC) illustrates huge diversity in dialogues, not just in the subject area and speaker background/register but also in mark-up and annotation practices. We urge the dialogue corpus research community to agree standards for transcription and markup format: this would help us, and others too.

Expanding AIML files using least frequent word and investigating how to incorporate corpus-derived linguistic annotation into an Elizabeth-style chatbot pattern file are the future direction of this research.

16. References

ALICE 2002 *A.L.I.C.E AI Foundation* , http://www.alicebot.org/ or http://alicebot.franz.com/

Abu Shawar B, Atwell E 2002 *A comparison between ALICE and Elizabeth chatbot systems*. School of Computing research report 2002.19, University of Leeds.

Millican P 2002 *Elizabeth's home page* http://www.etext.leeds.ac.uk/elizabeth

Mann W 2002 *Dialog Diversity Corpus* http://www-rcf.usc.edu/~billmann/diversity/DDivers-site.htm

MICASE 2002 *MICASE online homepage*, http://www.hti.umich.edu/m/micase/

Rayson Paul, 2002 "Matrix: A statistical method and software tool for linguistic analysis through corpus comparison", Unpublished PhD thesis, Department of Computing, Lancaster University

Ringenberg M 2002 CIRCLE's tutorial archive http://www.pitt.edu/~circle/Archive.htm

Athel 2002 Corpus of Spoken Professional American-English: description, http://www.athel.com/corpdes.html

CISD 2002 TRAINS Dialogue Corpus, http://www.cs.rochester.edu/research/cisd/resources/trains.html

Nelson G 2002 *International Corpus of English: the Singapore Corpus user manual*, http://www-rcf.usc.edu/~billmann/diversity/ICE-SIN_Manual.PDF

Mishler E 1985 *The discourse of medicine: dialectics of medical interviews,* New Jersey, Ablex http://www-rcf.usc.edu/~billmann/diversity/Tr.5.1a.gif

Atwell E 1996 Machine Learning from corpus resources for speech And handwriting recognition. In Thomas J, Short M (eds), *Using corpora for language research: studies in the honour of Geoffrey Leech*. Harlow, Longman, pp151-166.

Atwell E, Demetriou G, Hughes J, Schiffrin A, Souter C, Wilcock S 2000 A comparative evaluation of modern English corpus grammatical annotation schemes. *ICAME Journal* 24: 7-23.