CHAPTER FOUR

ANALYSIS AND FINDINGS

4.0 Introduction

This chapter presents the findings for this investigation with reference to the two research questions:

(1) How is the Interpersonal Metaphor realized in the data?

(i) Through the Metaphor of Mood

(ii) Through the Metaphor of Modality

(2) How does the Interpersonal Metaphor influence the interactiveness in the text?

Section 4.1 addresses Question (1)(i): “How is the Interpersonal Metaphor realized in the data through the Metaphor of Mood?”. Section 4.2 answers Question (1)(ii): “How is the Interpersonal Metaphor realized in the data through the Metaphor of Modality?” Question (2): “How does the Interpersonal Metaphor influence the interactiveness in text?” is answered throughout the analysis to avoid repetition and redundancy. Section 4.3 presents the distribution of Interpersonal Metaphor. Section 4.4 ends the chapter with a chapter summary.

It should also be noted that the metaphorical sentences drawn from the original text are not presented in the analysis in running order, but grouped according to the way they are analysed. The quantitative data is presented in bar charts to illustrate the distribution patterns of the Interpersonal Metaphor features.

4.1 Analysis of the Metaphor of Mood

The analysis of the Metaphor of Mood examines the semantic expansion within the Mood System. Findings related to the four semantic expansion domains analysed in the Metaphor of Mood are presented in Section 4.1.1 for metaphorical Declarative clauses, Sections 4.1.2 and 4.1.3 for metaphorical Interrogative clauses and Section 4.1.4 for metaphorical Imperative clauses. (For the full analysis of Metaphor of Mood, please refer to Appendix 2A and Appendix 2B).

4.1.1 Semantic Expansion: Declarative Mood

It is known that the Declarative Mood has statement as the speech function. It is found that the Declarative Mood clause has the potential of fulfilling another speech function, the Command.

4.1.1.1 Analysis of Declarative Clauses with Statement & Command

Table 4.1 shows Declarative Mood clauses with Statement & Question in Text 1 and Table 4.2 shows Declarative Mood clauses with Statement & Question in Text 2.

Legend	
Text 1: Textbook Chapter	R: Type of Realization
Text 2: Popular Text Chapter	M: Metaphorical clause (original text)
	C: Congruent clause

Table 4.1: Declarative Mood clauses with Statement & Question in Text 1

Eg.	Label	R	Declarative Mood Sentences
1	T1/INT/S4	M	Using a *loop statement*, you simply tell the computer to *print* a *string* a hundred times without having to code the *print statement* a hundred times.
		C	Using a *loop statement*, tell the computer to *print* a *string* a hundred times without having to code the *print statement* a hundred times.
2	T1/LCD/S2	M	If you want the user to decide whether to take another question, you can use a confirmation dialog to control the *loop*.
		C	If you want the user to decide whether to take another question, use a confirmation dialog to control the *loop*.
3	T1/LCD/S3	M	A confirmation dialog can be created using the following *statement*.
		C	Create a confirmation dialog using the following *statement.*
4	T1/LCD/S7	M	You can rewrite *Listing 4.1* using a confirmation dialog to let the user decide whether to continue the next question.
		C	Rewrite *Listing 4.1* using a confirmation dialog to let the user decide whether to continue the next question.
5	T1/LSV/S9	M	A sample run of the program is shown in *Figure 4.3*.
		C	See *Figure 4.3* for a sample run of the program
6	T1/DWL/S2	M	Its syntax is given below:
		C	See below for its syntax:
7	T1/DWL/S3	M	Its execution flow chart is shown in *Figure 4.4.*
		C	See *Figure 4.4* for its execution flow chart.
8	T1/DWL/S11	M	For example, you can rewrite the *while loop* in *Listing 4.2* using a *do-while loop*, as shown in *Listing 4.3*.
		C	For example, rewrite the *while loop* in *Listing 4.2* using a *do-while loop*, as shown in *Listing 4.3*.
9	T1/FL/S2	M	A *for loop* can be used to simplify the preceding *loop*.
		C	Use a *for loop* to simplify the preceding *loop*.
10	T1/FL/S3	M	In general, the syntax of a *for loop* is as shown below.
		C	In general, see below for the syntax of a *for loop.*
11	T1/FL/S5	M	The *for loop statement* starts with the keyword *for*, followed by a pair of parentheses enclosing *initial-action*, *loop-continuation-condition*, and *action-after-each-iteration*, and followed by the *loop* body enclosed inside braces.
		C	Start the *for loop statement* with the keyword *for,* enclose *initial-action, loop-continuation-condition* with a pair of parenthesis, and enclose the *loop* body inside braces.
12	T1/FL/S6	M	*Initial-action*, *loop-continuation-condition*, and *action-after-each-iteration* are separated by semicolons.
		C	Separate *Initial-action*, *loop-continuation-condition*, and *action-after-each-iteration* with semicolons.
13	T1/FL/S7	M	A *for loop* generally uses a variable to control how many times the *loop* body is executed and when the *loop* terminates.
		C	Use a variable in a *for loop* to control how many times the *loop* body is executed and when the *loop* terminates.
14	T1/FL/S8	M	This variable is referred to as a control variable.
		C	Refer this variable as a control variable.
15	T1/FL/S10	M	For example, the following *for loop* prints "Welcome to *Java*!" a hundred times.
		C	See the following to understand how *for loop* prints "Welcome to *Java*!" a hundred times.
16	T1/FL/S11	M	The flow chart of the *statement* is shown in *Figure 4.5(b)*.
		C	See *Figure 4.5(b)* for the flow chart of the *statement.*
17	T1/FL/S23	M	If there is only one *statement* in the *loop* body, as in this example, the

		C	braces can be omitted. If there is only one *statement* in the *loop* body, as in this example, omit the braces.
18	T1/FL/S24	M C	The control variable must always be declared inside the control structure of the *loop* or before the *loop*. Declare the control variable inside the control structure of the *loop* or before the *loop*.
19	T1/FL/S25	M C	The *loop* control variable is used only in the *loop*, and not elsewhere. Use the *loop* control variable only in the *loop*, and not elsewhere.
20	T1/FL/S28	M C	For example, you cannot reference *i* outside the *for loop* in the preceding code, because it is declared inside the *for loop*. Do not reference *i* outside the *for loop* in the preceding code, because it is declared inside the *for loop*.
21	T1/WLU/S1	M C	The *while loop* and *for loop* are called pre-test *loops* because the continuation condition is checked be*fo*re the *loop* body is executed. Refer the *while loop* and *for loop as* pre-test *loops* because the continuation condition is checked be*fo*re the *loop* body is executed.
22	T1/WLU/S2	M C	The *do-while loop* is called a post-test *loop* because the condition is checked after the *loop* body is executed. Refer The *do-while loop as* a post-test *loop* because the condition is checked after the *loop* body is executed
23	T1/WLU/S4	M C	That is, you can write a *loop* in any of these three forms. Write a *loop* in any of these three forms.
24	T1/WLU/S5	M C	For example, a *while loop* in (a) in the following figure can always be converted into the *for loop* in (b): See the following figure on how , a *while loop* in (a) can always be converted into the *for loop* in (b):
25	T1/WLU/S6	M C	A *for loop* in (a) in the next figure can generally be converted into the *while loop* in (b) except in certain special cases (see Review Question 4.12 *for* such a case): See the next figure on how A *for loop* in (a) can generally be converted into the *while loop* in (b) except in certain special cases (see Review Question 4.12 *for* such a case):
26	T1/WLU/S8	M C	In general, a *for loop* may be used if the number of repetitions is known, as, for example, when you need to *print* a message a hundred times. In general, use a *for loop* used if the number of repetitions is known, as, for example, when you need to *print* a message a hundred times.
27	T1/WLU/S9	M C	A *while loop* may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0. Use a *while loop* if the number of repetitions is not known, as in the case of reading the numbers until the input is 0.
28	T1/WLU/S10	M C	A *do-while loop* can be used to replace a *while loop* if the *loop* body has to be executed before the continuation condition is tested. Use a *do-while loop* to replace a *while loop* if the *loop* body has to be executed before the continuation condition is tested.
29	T1/NL/S3	M C	*Listing 4.4* presents a program that uses *nested for loops* to *print* a multiplication table, as shown in *Figure 4.6*. See Listing 4.4 that presents a program that uses *nested for loops* to *print* a multiplication table, as shown in *Figure 4.6*
30	T1/MNE/S2	M C	This section discusses how to minimize such errors through an example. Read this section that discusses how to minimize such errors through an example.
31	T1/MNE/S3	M C	*Listing 4.5* presents an example that *sum*s a series that starts with *0.01* and ends with *1.0.* See *Listing 4.5* that presents an example that *sum*s a series that starts

			with *0.01* and ends with *1.0.*
32	T1/MNE/S10	M C	From this example, you can see that a control variable can be a *float* type. From this example, see that a control variable can be a *float* type.
33	T1/MNE/S14	M C	If you change *float* in the program to *double* as follows, you should see a slight improvement in precision because a *double* variable takes sixty-four *bits*, whereas a *float* variable takes thirty-two *bits*. If you change *float* in the program to *double* as follows, notice a slight improvement in precision because a *double* variable takes sixty-four *bits*, whereas a *float* variable takes thirty-two *bits*.
34	T1/MNE/S15	M C	However, you will be stunned to see that the result is actually *49.50000000000003.* However, see that the result is actually *49.50000000000003.*
35	T1/MNE/S22	M C	Minimizing errors by processing large numbers first. Minimize errors by processing large numbers first.
36	T1/MNE/S23	M C	Using an integer *count* to ensure that all the numbers are processed. Use an integer *count* to ensure that all the numbers are processed.
37	T1/MNE/S26	M C	Here is the new *loop*. See here for the new *loop.*
38	T1/CS/S4	M C	For this reason, this section presents three additional examples of how to solve problems using *loops*. For this reason, read this section that presents three additional examples of how to solve problems using *loops*.
39	T1/FGCD/S1	M C	This section presents a program that prompts the user to enter two positive integers and finds their greatest common divisor. Read this section that presents a program that prompts the user to enter two positive integers and finds their greatest common divisor.
40	T1/FGCD/S4	M C	How do you find the greatest common divisor? Find the greatest common divisor
41	T1/FGCD/S7	M C	So you can check whether *k (*for *k = 2, 3, 4* and so on) is a common divisor for *n1* and *n2*, until *k* is greater than *n1* or *n2.* Check whether *k (*for *k = 2, 3, 4* and so on) is a common divisor for *n1* and *n2*, until *k* is greater than *n1* or *n2.*
42	T1/FGCD/S12	M C	The idea can be translated into the following *loop*: Translate the idea into the following *loop*
43	T1/FGCD/S15	M C	How did you write this program? Describe how you wrote this program.
44	T1/FGCD/S18	M C	It is important to think before you type. Think before you type.
45	T1/FGCD/S22	M C	For example, you could use a *for loop* to rewrite the code as follows: For example, use a *for loop* to rewrite the code as follows:
46	T1/FGCD/S26	M C	A more efficient solution is to use the classic Euclidean algorithm. For a more efficient solution, use the classic Euclidean algorithm.
47	T1/FSA/S6	M C	This section writes a program that finds the minimum amount of sales you have to generate in order to make *$30,000.* Read this section that writes a program that finds the minimum amount of sales you have to generate in order to make *$30,000.*
48	T1/FSA/S8	M C	What is the sales amount *for* a *$25,000* commission? Find the sales amount *for* a *$25,000* commission.
49	T1/FSA/S9	M C	If you know the sales amount, the commission can be computed as follows. If you know the sales amount, compute the commission as follows.
50	T1/FSA/S10	M	This suggests that you can try to find the *salesAmount* to match a given commission through incremental approximation. Try to find the *salesAmount* to match a given commission through

		C	incremental approximation.
51	T1/FSA/S15	M C	You can write a *loop* and let a computer execute it painlessly. Write a *loop* and let a computer execute it painlessly.
52	T1/FSA/S16	M C	The idea can be translated into the following *loop*: Translate the idea into the following loop:
53	T1/FSA/S20	M C	In *Exercise 4.17*, you will rewrite this program to let the user enter *COMMISSION_SOUGHT* dynamically from an input dialog. In *Exercise 4.17*, rewrite this program to let the user enter *COMMISSION_SOUGHT* dynamically from an input dialog.
54	T1/FSA/S21	M C	You can improve the performance of this program by estimating a higher *INITIAL_SALES_AMOUNT (e.g., 25000).* Improve the performance of this program by estimating a higher *INITIAL_SALES_AMOUNT (e.g., 25000).*
55	T1/FSA/S22	M C	What is wrong if *salesAmount* is incremented after the commission is computed, as follows? Find the mistake if *salesAmount* is incremented after the commission is computed, as follows?
56	T1/DPN/S1	M C	This section presents a program that prompts the user to enter an integer from *1* to *15* and displays a pyramid. Read this section that presents a program that prompts the user to enter an integer from *1* to *15* and displays a pyramid.
57	T1/DPN/S12	M C	You can use an outer *loop* to control the lines. Use an outer *loop* to control the lines.
58	T1/DPN/S14	M C	You can use three separate inner *loops* to *print* each part. Use three separate inner *loops* to *print* each part.
59	T1/DPN/S15	M C	Here is the algorithm for the problem. Read the algorithm for the problem.
60	T1/DPN/S16	M C	The complete program is given in *Listing 4.8.* See the complete program in *Listing 4.8.*
61	T1/KBC/S1	M C	Two *statements*, *break* and *continue*, can be used in *loop statements* to provide the *loop* with additional control. Use the two *statements*, *break* and *continue* in *loop statements* to provide the *loop* with additional control.
62	T1/KBC/S3	M C	It is generally used with an *if statement*. Use it with an *if statement.*
63	T1/KBC/S6	M C	This keyword is generally used with an *if statement*. Use this keyword with an *if statement.*
64	T1/KBC/S8	M C	You can also use *break* and *continue* in a *loop*. Use *break* and *continue* in a *loop*.
65	T1/KBC/S20	M C	The output of the program is shown in *Figure 4.12(a).* See the output of the program in *Figure 4.12(a).*

Table 4.2: Declarative Mood clauses with Statement & Question in Text 2

Eg.	Label	R	Declarative Mood Sentences
1	T2/LEO/S4	M C	Congratulations, you just experienced a perfect example of a verbal *loop*! Congratulations, experience a perfect example of a verbal *loop!*
2	T2/GRL/S1	M C	Let's pretend NASA used *Java* applets to control the launch of the space shuttle. Pretend NASA used *Java* applets to control the launch of the space shuttle.
3	T2/GRL/S6	M C	Following is code to perform the launch sequence without the use of a *loop*. See the following for the code to perform the launch sequence without the use of a *loop*.

4	T2/GRL/S7	M C	And now the *loop* version: And now see the *loop* version:
5	T2/GRL/S9	M C C	You probably wonder exactly how the *loop* code works. You probably wonder, how exactly does a *loop* work? Think of how does the loop code work.
6	T2/GRL/S14	M C	The *InitializationExpression* is used to initialize a *loop* control variable. Use the *InitializationExpression* to initialize a *loop* control variable.
7	T2/GRL/S17	M C	Let's take a look at the NASA launch sequence code again to make some sense of this stuff. Take a look at the NASA launch sequence code again to make some sense of this stuff.
8	T2/GRL/S19	M C	This is the code you use to prime the *loop* and get it ready. Use this code to prime the *loop* and get it ready.
9	T2/GRL/S30	M C	You can feel safe and secure knowing that I'm not running *for* president or trying to help you visualize my answer to global trade. Be safe and secure knowing that I'm not running *for* president or trying to help you visualize my answer to global trade.
10	T2/GRL/S37	M C	If you recall, this grouping of *statements* is known as a *compound statement* and was used in the previous chapter when dealing with *if-else* branches. Recall that this grouping of *statements* is known as a *compound statement* and was used in the previous chapter when dealing with *if-else* branches.
11	T2/GRL/S38	M C	Following is an example of a *for loop* with a *compound statement:* See the following for an example of a *for loop* with a *compound statement:*
12	T2/GRL/S42	M C	It is necessary to subtract 1 in this case because all *Java* array indexes start with 0, which means they are zero based. Subtract 1 in this case because all *Java* array indexes start with 0, which means they are zero based.
13	T2/LJLW/S6	M C	Following is the syntax *for* the *while loop*, which should make its usage a little more clear: See the following for the syntax for the *while loop*, which should make its usage a little more clear:
14	T2/LJLW/S10	M C	Because the *while loop* has no *step expression*, it is important to make sure that the *Statement* somehow impacts the *LoopCondition*. Because the *while loop* has no *step expression*, make sure that the *Statement* somehow impacts the *LoopCondition*.
15	T2/LJLW/S12	M C	Following is a simple example of an infinite *while loop*: See the following for a simple example of an infinite *while loop*:
16	T2/LJLW/S18	M C	You can think of the *while loop* as a more general *for loop*. Think of the *while loop* as a more general *for loop*.
17	T2/LJLW/S29	M C	This code demonstrates how a *while loop* could be used to ask a question and patiently wait for the correct answer. See this code that demonstrates how a *while loop* could be used to ask a question and patiently wait for the correct answer.
18	T2/LJLW/S38	M C	If you aren't completely satisfied with *while loops*, however, there is one other option. If you aren't completely satisfied with *while loops*, however, consider one other option.
19	T2/TDNTD/S2	M C	Because you're becoming pretty *loop* savvy, I'll show you the syntax *for* the *do-while loop* first and see if you can figure out how it works. Because you're becoming pretty *loop* savvy, take note on the syntax *for* the *do-while loop* first and see if you can figure out how it works.
20	T2/AC/S4	M	In this section you use your knowledge of *loops* to build a "countdown"

		C	applet that counts down from ten to one and then navigates to a new Web page. In this section, use your knowledge of *loops* to build a "countdown" applet that counts down from ten to one and then navigates to a new Web page.
21	T2/AC/S5	M C	The following figure shows the *Countdown* applet in action. See the following figure that shows the *Countdown* applet in action.
22	T2/AC/S7	M C	As an example, what web site could be better than NASA's to demonstrate how this applet works? As an example, think of what web site that could be better than NASA's to demonstrate how this applet works.
23	T2/AC/S8	M C	Following is NASA's Web site, to which the *Countdown* applet will take you after it finishes its countdown. See the following for NASA's Web site to which the *Countdown* applet will take you after it finishes its countdown.
24	T2/AC/S11	M C	The main thing on which I want you to focus is the page parameter, which is defined as: Focus on the page parameter, which is defined as:
25	T2/AC/S16	M C	They enable you to customize the function of applets without doing any real programming! Customize the function of applets without doing any real programming!
26	T2/AC/S21	M C	Following is the *run() method* in the *Countdown* applet class, which forms the heart of the applet: See the following for the *run() method* in the *Countdown* applet class, which forms the heart of the applet:
27	T2/AC/S25	M C	As you can see, the *for loop* counts down from 10 to 1 just like the *Countdown* code you saw earlier in the chapter. See that the *for loop* counts down from 10 to 1 just like the *Countdown* code you saw earlier in the chapter.
28	T2/AC/S32	M C	I'll explain *exceptions* as you encounter them throughout the book. Be ready for explanation of *exceptions* as you encounter them throughout the book.
29	T2/AC/S33	M C	The complete source code for the *countdow*n applet is as follows. See the complete source code for the *countdow*n applet as follows.
30	T2/BA/S1	M C	If you recall from the previous chapter, each case section of a *switch* branch ends with a *break statement*. Recall from the previous chapter, each case section of a *switch* branch ends with a *break statement*.
31	T2/BA/S2	M C	Following is an example to recap: See the following for an example to recap:
32	T2/BA/S6	M C	Following is an example of circumventing an infinite *loop* with a *break statement*: See the following for an example of circumventing an infinite *loop* with a *break statement*:
33	T2/BA/S13	M C	The following example shows how a *continue statement* can be used to *print* only the even numbers between 1 and 100: See the following example that shows how a *continue statement* can be used to *print* only the even numbers between 1 and 100:
34	T2/BA/S19	M C	The example code exploits this characteristic of even and odd numbers to skip to the next iteration bypasses the *println()* call, which prevents odd numbers from being printed. See that the example code exploits this characteristic of even and odd numbers to skip to the next iteration bypasses the *println()* call, which prevents odd numbers from being printed.
35	T2/LNK/S8	M	A *for loop* is used to repeat a section of code a given number of

		C	iterations. Use a *for loop* to repeat a section of code a given number of iterations.
36	T2/LNK/S11	M C	The *break statement* is used to break out of a *loop* regardless of the *loop* condition. Use the *break statement to break out of a loop* regardless of the *loop* condition.
37	T2/LNK/S12	M C	The *continue statement* is used to skip to the next iteration of a *loop*. Use the *continue statement* to skip to the next iteration of a *loop*.

4.1.1.2 Findings and Distribution of the Declarative Clauses with Statement & Command

The metaphorical clauses are analysed according to the semantic function and lexicogrammatical features as discussed below.

(a) <u>Semantic Function</u>

It is understood that Declarative clauses fulfill Statement as speech function. The clause is meant to share information with the readers. This is in agreement with Azirah's (1996) findings on the high usage of Declarative to discuss issues in medical research. A metaphorical Declarative clause opens up the potential for the clause to function as a command. Therefore, the clause not only gives information and facts to the readers, but also initiates an action to be carried out by the reader. The findings show that the semantic expansion of Declarative has several semantic functions as listed below:

a(i) <u>Functioning as an advice</u>

In this semantic expansion, the metaphorical clause also acts as a form of advice for the readers so that the text can be more accessible to the readers. According to Wong's (2009) study on self-help texts, the writer uses Mood to express different Interpersonal meanings. High Declarative indicate the emphasis on providing

information whereas Imperative provide instruction to solve problems as advice and to enhance persuasiveness. A metaphorical Declarative clause fulfills both functions such as in Example 1 in Table 4.1, “Using a loop statement, you simply tell the computer to…”. The congruent form is in the form of a command “Using a loop statement, tell the computer to…” draws the reader to respond by following the advice of “telling the computer”.

a(ii) Providing text direction

The metaphorical clause in the original text guides the reader with explanation like statements, especially in introducing figures and charts such as in Example 5 in Table 4.1 “A sample run of the program is shown in Figure 4.3”. The congruent form, “See the sample run of the program in Figure 4.3.” requests for readers to take the action of “seeing”. This will guide the readers in the development of the text. Knowing the direction of the text enables the readers to understand the text more effectively.

(b) Lexicogrammatical Features

The lexicogrammatical features found in the metaphorical clauses are:

b(i) Grammatical person ‘you’

While Ho (2004) discusses that using grammatical person ‘you’ will result in a more interactive text, Wales (2006) states that ‘you’ may appear to be impersonal in most cases where the grammatical pronoun is being used to address a general audience. In the two texts, ‘you’ is being used to address the reader in general terms such as in Example 4, Table 4.1, “You can rewrite…” Hence, ‘you’ does not contribute in making the text more personal. However, there are instances where ‘you’ is used to

create a more personal text. This is discussed on Page 67 where ‘you’ is used in Question forms.

b(ii) Ellipsis

Declaratives may contain ellipted clause such as in Example 1 in Table 4.2, “Congratulations, you just experienced a perfect example of a verbal *loop*”. ‘Congratulations’ is an ellipted clause of descriptive statement ‘(let me offer my) congratulations’ (Langacker, 2008:475). The author reenacts the style of speaker-hearer interaction with ellipsis, promoting a closer relationship between the two participants.

b(iii)‘Objectifying’ sentences through ‘relational’ clauses

The “objectivisation” of a clause gives the writer an authority to establish a fact that cannot be refuted or argued (Scheibman, 2002). Ravelli & Ellis (2004:45) adds that an objectified sentence concretises the clause such that there is no room for readers to negotiate the meaning of the clause, but to accept the clause as an advice given by the author with the role of ‘consultant’ in the text. Example 44 in Table 4.1: ‘It is important to think before you type’ is objectified from the congruent clause “Think before you type”.

b(iv) Passivization of clause

Similar to the purpose of objectifying a clause, Lunt (2008) and Ravelli & Ellis (2004) observe that a passivised clause such as in Example 3 in Table 4.1, “A confirmation dialog can be created…” enables the Agent (a person involved in the text) to be removed from the clause. Ravelli and Ellis (2004) further elaborate that a passive construction hides the actor from the context when it does not become

totally clear who the doer might be. This abstraction leads the reader to construe that the hidden actor is a concrete person since the reader cannot refute or argue with the abstraction. Hence, the passive voice can be regarded as advice given to the reader by the author, without revealing the identity (Ravelli and Ellis, 2004).

b(v) Reference

The feature of reference is used when "reader has to retrieve the identity of what is being talked about by referring to another expression in the immediate context" (Baker, 1992:181). According to Halliday and Hasan (1976:31) "reference lies in the continuity of reference, whereby the same thing enters into the discourse the second time". In the sentence above, 'this' is used to establish links between expressions in text. Baker elaborates that reference is a linguistic device which allows readers "to trace participants, entities and events" (Baker, 1992:181).

Example 63 in Table 4.1 has the feature of reference from the preceding sentence [T1/KBC/S4]: *"Continue* only ends the current iteration".

In the Example 63 in Table 4.1: "**This** keyword is generally used with an *if statement",* **'this'** refers to the keyword '*continue'.* It serves as a grammatical constituent that function as a point of reference for readers. Words like 'this', 'that', 'these' and 'those' are referred to as 'deictics' (Carter & Goddard, 2003:128).

b(vi) Marked Theme

The author uses marked theme to place emphasis on the message that he/she wants to appear as informative (Granger, 2003, Claridge, 2000 & Halliday, 1967) such as in Example 1 in Table 4.1, "Using a *loop statement,* you simply tell..." The

marked theme is “Using a *loop statement”* which highlights the focus of information, “*loop statement”.*

(c) Distribution Patterns of Declarative Clauses with Statement & Command

The distribution patterns of Declarative clauses with Statement and Command are shown in Table 4.3.

Table 4.3: Distribution Patterns of Declarative Clauses with Statement and Command

Text	No. of Occurences	Total No. of Sentences with the Metaphor of Mood	Percentage	Calculation of Percentage
1	65	79	82.28%	(65/79) x 100%
2	37	64	57.81%	(37/64) x 100%

The distribution patterns of Declarative clauses with Statement and Command are illustrated in Figure 4.1.

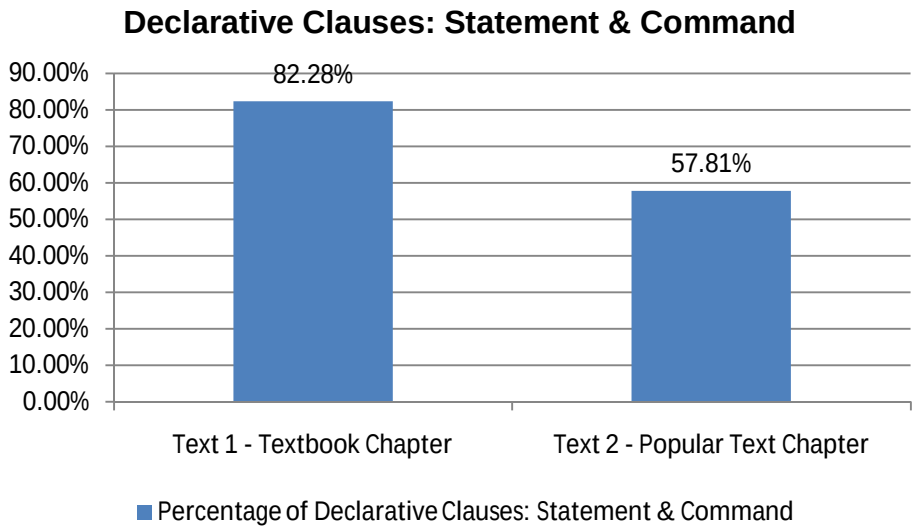

Figure 4.1: Distribution Patterns of Declarative Clauses with Statement and Command

It is shown that Text 1 has a higher percentage of Declarative clauses with Statement and Command as compared to Text 2. This shows that Text 1 contains 5.47% more metaphorical clauses compared to Text 2. The statements provide advice and

direction to the readers. This is in line with the function of a textbook that provides pedagogical guidance for students to undertake the subject matter.

Although Text 1 shows a higher occurrence of Declarative metaphorical clauses than Text 2, it cannot be concluded in this stage that Text 1 is more interactive than Text 2. To determine whether Text 1 or Text 2 is more interactive as a whole, the occurrences of all Mood choices, including Interrogative and Imperative Mood choices need to be taken into consideration as well. The analysis of Interrogative Mood Choices is shown in Section 4.1.2 and Imperative Mood Choices is shown in Section 4.1.3.

4.1.2 Semantic Expansion: Interrogative Mood

The Interrogative Mood has question as the speech function. It is found that the Interrogative Mood clause has the potential of fulfilling two other speech functions: Statement and Command.

The analysis of the Interrogative Mood with Question and Statement is discussed in Section 4.1.2.1 and the analysis of Interrogative Mood with Question and Command is discussed in Section 4.1.2.1.

4.1.2.1 Analysis of the Interrogative Mood Clauses with Question & Statement

Table 4.4 shows Interrogatives with Question and Statement from Text 1 and Table 4.5 shows Interrogatives with Question and Statement from Text 2.

Legend	
Text 1: Textbook Chapter	R: Type of Realization
Text 2: Popular Text Chapter	M: Metaphorical clause (original text)
	C: Congruent clause

Table 4.4: The Interrogative Mood clauses with Question & Statement in Text 1

Eg.	Label	R	Sentence
1	T1/LSV/S5	M	Do you need to declare a new variable for each input value?
	T1/LSV/S6	M C	No. There is no need to declare a new variable for each input.
2	T1/MNE/S16	M C	What went wrong? There is a mistake.
3	T1/FGCD/S16	M	Did you immediately begin to write the code?
	T1/FGCD/S17	M C	No. You didn't immediately begin to write the code.

Table 4.5: Interrogative Mood clauses with Question & Statement in Text 2

Eg.	Label	R	Sentence
1	T2/LEO/S1	M C	Have you ever been talking to someone and it seems like he or she is saying the same thing over and over? There is probably a time when you talk to someone and it seems like he or she is saying the same thing over and over.
3	T2/GRL/S8	M C	See what I mean about tightening up the code? This is what I mean about tightening up the code.
4	T2/LJLW/S24	M C	If a *for loop* can do everything a *while loop* can and in a more organized way, then why do we need *while loops*? Even when a *for loop* can do everything a *while loop* can and in a more organized way, we still need *while loops.*
5	T2/TDNTD/S3	M C	Give up? If you can't figure it out, the answer is as follows:
6	T2/TDNTD/S5	M C	Why is this necessary? This is necessary.
7	T2/TDNTD/S12	M C	What do I mean by this? The following describes what I mean by this.
8	T2/AC/S1	M C	Have you ever visited a Web page that directed you to another page, but informed you that if you waited a few second sit would automatically take you there? There is probably a time when you visit a Web page that directed you to another page, but informed you that if you waited a few second sit would automatically take you there.
9	T2/BA/S14	M C	Having trouble seeing how this one works? You might have trouble seeing how this one works.

4.1.2.2 Findings and Distribution of Interrogative Mood Clauses with Question & Statement

The analysis of clauses is based on the semantic function and lexicogrammatical features.

(a) Semantic Function

The semantic function of Interrogative Mood clauses with Question and Statement include the author strengthening their statements using rhetorical questions and answering their own questions, as well as providing text direction.

a(i) Rhetorical Questions

Instead of having the purpose of drawing replies from the readers, rhetorical question asserts the position of the author (Archer, 2005) such as in Example 6 in Table 4.5, “Why is this necessary?” The author does not expect an answer from the reader because the “answer” to this question is “self evident” (Archer, 2005:27). With this metaphorical clause, the author is making an indirect Statement saying that “This is necessary”. As observed by Archer (2005), rhetorical questions are used as a politeness strategy because it softens criticisms since facts and statements are made indirectly. It is also a device for authors to make stronger statements (Frank, 1990) as compared to writing it in the form of a Declarative.

a(ii) Author answering his/her own question

The author assumes the position of the reader to answer to his own question in the text such as in Example 1 in Table 4.4, “Do you need to declare…?” to which the author replies to his own question “No.” This self-answered question states the fact that “There is no need to declare…” Also, posing a question in text encourages the readers to participate actively in text. To keep readers on the right track in understanding the text, authors provide the correct answer immediately after the question.

a(iii) Providing text direction

Through Interrogatives, the author provides direction through text development such as in Example 5, Table 4.5, “Give up?” The author means “If you can’t figure this out, the answer is as follows.” This semantic feature is used to guide the readers according to the flow of the text development.

(b) Lexicogrammatical Features

The lexicogrammatical features of Interrogative Mood clauses with Question and Statement include:

b(i) Grammatical person ‘you’

As mentioned earlier in Page 60 and Page 61, ‘you’ is used as a device to address the readers in general terms, instead of directly addressing individual readers. In Interrogative clauses, the questions with grammatical person ‘you’ is supported by an answer given by the author himself such as in Examples 3 and 4 in Table 4.4, “Did you immediately begin to write the code?” “No.”

b(ii) Vocative Adjuncts

Vocative Adjuncts are “Yes” or “No”. According to Eggins (2004:162), vocative adjuncts control the discourse by “designating a likely ‘next speaker’”. This means that, after the author asks “Did you immediately begin to write the code?” the ‘next likely’ speaker would be the reader, since the reader is expected to reply to that question with a “Yes” or a “No”. The usage of vocative indicates an interactive text. According to Sim’s (2008) findings, a low usage of vocative shows a non-interactive text since there is no direct personal confrontation between reader and writer.

(c) Distribution Patterns of Interrogative Clauses with Question & Statement

The distribution patterns of Interrogative Clauses with Question & Statement are shown in Table 4.6.

Table 4.6: Distribution Patterns of Interrogative Clauses with Question & Statement

Text	No. of Occurences	Total No. of Sentences with the Metaphor of Mood	Percentage	Calculation of Percentage
1	3	79	4.69%	(3/79) x 100%
2	9	64	14.06%	(9/64) x 100%

The distribution patterns of Interrogative Clauses with Question & Statement are illustrated in Figure 4.2.

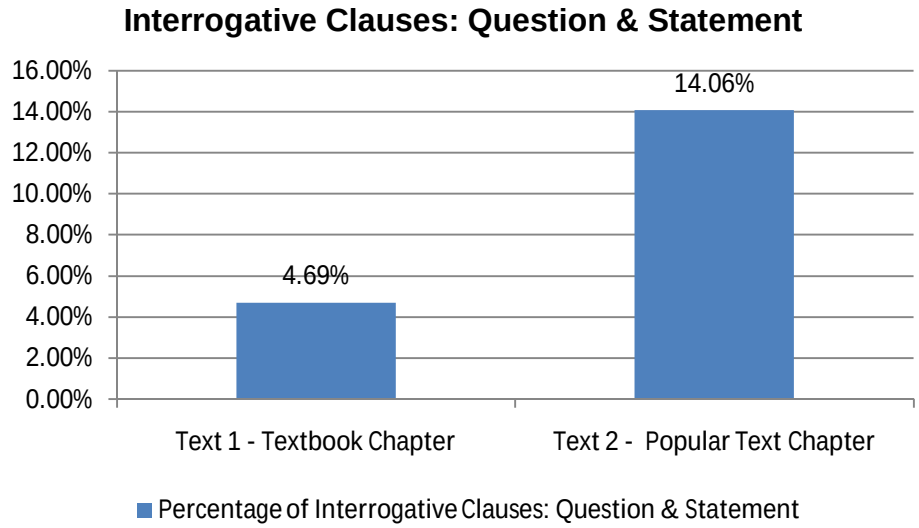

Figure 4.2: Distribution Patterns of Interrogative Clauses: Question & Statement

Figure 4.2 shows that Text 2 has 9.37% more Interrogative clauses with Question and Statement compared to Text 1. This shows that Text 2 have instances of the author strengthening his statement with a higher number of rhetorical questions. The author asks questions without expecting answers. The rhetorical questions are related to the politeness stance. Hence, it can be concluded that Text 2 attempts to make the text more persuasive and at the same time, friendlier towards the readers by the use of metaphorical Interrogative clauses.

4.1.2.3 Analysis of Interrogative Mood Clauses with Question & Command

Table 4.7 shows Interrogative Mood Clauses realising Question and Command in Text 1 and Table 4.8 shows Interrogative Mood Clauses realising Question and Command in Text 2.

Legend

Text 1: Textbook Chapter
Text 2: Popular Text Chapter

R: Type of Realization
M: Metaphorical clause (original text)
C: Congruent clause

Table 4.7: Interrogative Mood clauses with Question & Command in Text 1

Eg.	Label	R	Sentence
1	T1/FGCD/S31	M C	Can you find the reason? Find the reason.

Table 4.8: Interrogative Mood clauses with Question & Statement in Text 2

Eg.	Label	R	Sentence
1	T2/GRL/S2	M C	Any ideas on how controllers would initiate the launch sequence? Think of how controllers would initiate the launch sequence.

4.1.2.4 Findings and Distribution of Interrogative Mood Clauses with Question & Command

The clauses are analysed based on the semantic function.

(a) Semantic Function

It is found that Interrogatives with Question and Command has this semantic function:

a(i) Engaging the reader

The author requests the reader to carry out an action in the form of Interrogatives such as in Example 1 in Table 4.7, “Can you find the reason?” The use of the indirect command in congruent form, “Find the reason”, compels the reader to carry out the action of “finding” according to the instructions of the author.

a(ii) Ellipsis

Example 1 in Table 4.8, “Any ideas on how…?” is an ellipted clause which comes from “(do you have) any ideas on how…?” The omission of “do you have” enables the emphasis to remain on “ideas”. Fry (2003:84) observes that one of the purposes of ellipsis is for emphasis to convey the message without having to state the obvious and for the “packaging of the message.”

(b) Distribution Patterns of Interrogative Clauses with Question & Command

The distribution patterns of Interrogative Clauses with Question & Command are shown in Table 4.9.

Table 4.9: Distribution Patterns of Interrogative Clauses with Question & Command

Text	No. of Occurences	Total No. of Sentences with the Metaphor of Mood	Percentage	Calculation of Percentage
1	1	79	1.27%	(1/79) x 100%
2	1	64	1.56%	(1/64) x 100%

The distribution patterns of Interrogative Clauses with Question & Command are illustrated in Figure 4.3.

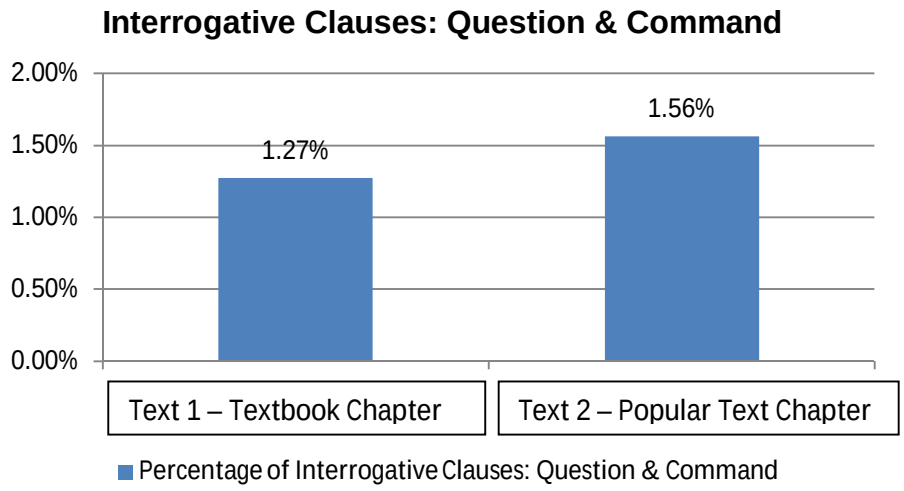

Figure 4.3: Distribution Patterns of Interrogative Clauses: Question & Command

Both Text 1 and Text 2 have only one occurrence of the Interrogative clause realising Question and Command. The clause directs readers to carry out an action with the Interrogative to enhance reader's participation in text.

4.1.3 Semantic Expansion: Imperative Mood

The Imperative Mood has command as the speech function. It is found that the Imperative Mood clause has the potential of fulfilling another speech function: Statement.

4.1.3.1 Analysis of Imperative Mood Clauses with Command & Statement

Table 4.10 shows Imperative Mood Clauses realising Command & Statement in Text 1 and Table 4.11 shows Imperative Mood Clauses realising Command & Statement in Text 2.

Legend

Text 1: Textbook Chapter
Text 2: Popular Text Chapter

R: Type of Realization
M: Metaphorical clause (original text)
C: Congruent clause

Table 4.10: Imperative Mood Clauses with Command & Statement in Text 1

Eg.	Label	R	Sentence
1	T1/LSV/S7	M	Just use one variable named *data* (*line* 9) to store the input value and use a variable named *sum* (*line* 12) to store the total.
		C	One variable named *data* (*line* 9) is used to store the input value and a variable named *sum* (*line* 12) is used to store the total
2	T1/LSV/S13	M	Note that if the first input read is 0, the *loop* body never executes, and the resulting *sum* is 0.
		C	If the first input read is 0, the *loop* body never executes, and the resulting *sum* is 0.
3	T1/WLU/S7	M	Use the *loop statement* that is most intuitive and comfortable for you.
		C	You can use the *loop statement* that is most intuitive and comfortable for you.
4	T1/MNE/S24	M	To minimize errors, add numbers from *1.0, 0.99,* down to *0.1,* as follows:
		C	To minimize errors, you can add numbers from *1.0, 0.99,* down to *0.1,* as follows:
5	T1/MNE/S25	M	To ensure that all the items are added to *sum*, use an integer variable to *count* the items.
		C	To minimize errors, you can use an integer variable to *count* the items.
6	T1/FGCD/S5	M	Let the two input integers be *n1* and *n2.*
		C	You can let the two input integers be *n1* and *n2.*

7	T1/FGCD/S8	M C	Store the common divisor in a variable named *gcd.* You can store the common divisor in a variable named *gcd.*
8	T1/FGCD/S20	M C	Once you have a logical solution, type the code to translate the solution into a *Java* program. Once you have a logical solution, you can type the code to translate the solution into a *Java* program.
9	T1/FGCD/S27	M C	See *http://www.cut-the-knot.org/blue/Euclid.shtml* for more information. You can see *http://www.cut-the-knot.org/blue/Euclid.shtml* for more information.
10	T1/FGCD/S32	M C	See *Review Question 4.9* for the answer. You can see *Review Question 4.9* for the answer.

Table 4.11: Imperative Mood Clauses with Command & Statement in Text 2

Eg.	Label	R	Imperative Mood Sentences
1	T2/GRL/S29	M C	Just ask Ross Perot, who isn't a *Java* programmer but who nonetheless relied on diagrams and illustrations to help us grasp his big plans for the presidency. You can ask Ross Perot, who isn't a *Java* programmer but who nonetheless relied on diagrams and illustrations to help us grasp his big plans for the presidency.
2	T2/GRL/S32	M C	To help you visualize the *looping* process, take a look at the following figure. To help you visualize the *looping* process, you can take a look at the following figure.
3	T2/GRL/S33	M C	Notice in the figure that *Statement 1* and *Statement 2* will be repeatedly executed as long as the *loop* condition is true. You can notice in the figure that *Statement 1* and *Statement 2* will be repeatedly executed as long as the *loop* condition is true.
4	T2/GRL/S40	M C	Notice that the *loop counter (i)* is used as the index *(i-1)* into the squares array. You can notice that the *loop counter (i)* is used as the index *(i-1)* into the squares array.
5	T2/GRL/S44	M C	Rest assured I would be the first to tell you if they did! You can be rest assured I would be the first to tell you if they did!
6	T2/LJLW/S19	M C	To understand what I mean by this, check out the following code. To understand what I mean by this, you can check out the following code.
7	T2/LJLW/S28	M C	Consider the following example: You can consider the following example:
8	T2/LJLW/S33	M C	Just assume that they somehow present the user with a question, retrieve an answer, and then judge the correctness of the answer. You can assume that they somehow present the user with a question, retrieve an answer, and then judge the correctness of the answer.
9	T2/TDNTD/S8	M C	Let's take a look at the question and answer example implemented using a *do-while loop*: You can take a look at the question and answer example implemented using a *do-while loop*:
10	T2/AC/S9	M C	To understand how the *Countdown* applet works, let's first take a look at the *Countdown. Html* Web page that contains the embedded applet. To understand how the *Countdown* applet works, you can first take a look at the *Countdown. Html* Web page that contains the embedded applet.
11	T2/AC/S12	M C	Notice that the value of the page parameter is set to http://www.nasa.gov, which is the URL of NASA's Web site. You can notice that the value of the page parameter is set to http://www.nasa.gov, which is the URL of NASA's Web site.
12	T2/AC/S17	M	Let's move on to the actual code required of the countdown applet.

		C	You can move on to the actual code required of the countdown applet.
13	T2/AC/S23	M C	Try not to get intimidated by any code that doesn't look familiar. You must try not to get intimidated by any code that doesn't look familiar.
14	T2/AC/S24	M C	Just concentrate on the *loop* code. You can just concentrate on the *loop* code.
15	T2/BA/S15	M C	Think back to the *modulus operator* (%), which returns the remainder of a division. You can think back to the *modulus operator* (%), which returns the remainder of a division.
16	T2/BA/S16	M C	Now consider what the remainder of a division by 2 yields for even and odd numbers. Now, you can consider what the remainder of a division by 2 yields for even and odd numbers.
17	T2/LNK/S6	M C	Let's go over the main points you learned about *loops* in this chapter. You can go over the main points you learned about *loops* in this chapter.

4.1.3.2 Findings and Distribution of Imperative Mood Clauses with Command & Statement

The clauses are analysed based on the semantic functions and lexicogrammatical features.

(a) Semantic Function

It is found that the Imperative Mood clauses with Command and Statement have these semantic functions.

a(i) Functioning as an advice

Martin (1992) observes that the Imperative has so far been regarded as the typical realization of a command. However, Lassen (2003) contrasts his views by stating that Imperatives can be used in the text to provide advice, especially in technical manuals. Through Lassen's (2003:41) analysis, Imperatives are used as "advice offered to be acted voluntarily" such as in Example 6 in Table 4.10, "Let the two input integers be *n1* and *n2*". As suggested by Lassen, in an utterance, it could be "you can let the two input integers be *n1* and *n2"*. He observes that the Process (Let) still comes before the Goal (*n1* and *n2).* In summary, Lassen (2003:41) terms this

phenomenon as a “semantic cross coupling” where Imperatives can be realized by either a Command or a Statement to fulfill the function of an advice.

(b) Lexicogrammatical Feature

It is found that the Imperative Mood clauses with Command and Statement have this lexicogrammatical feature.

b(i) Imperative particle “Let’s”

The clause element ‘Let’s’ is noted by (Downing & Locke, 2006:190) to be an ‘Imperative particle’ suggesting a joint action involving the reader and the writer such as in Example 12 in Table 4.11, “Let’s move on to…” This promotes the interaction between the two participants in text.

(c) Distribution Patterns of Imperative Clauses with Command & Statement

The distribution patterns of Imperative Clauses with Command & Statement are shown in Table 4.12.

Table 4.12: Distribution Patterns of Imperative Clauses with Command & Statement

Text	No. of Occurences	Total No. of Sentences with the Metaphor of Mood	Percentage	Calculation of Percentage
1	10	79	12.66%	(10/79) x 100%
2	17	64	26.56%	(17/64) x 100%

The distribution patterns of Imperative Clauses with Command & Statement are illustrated in Figure 4.4.

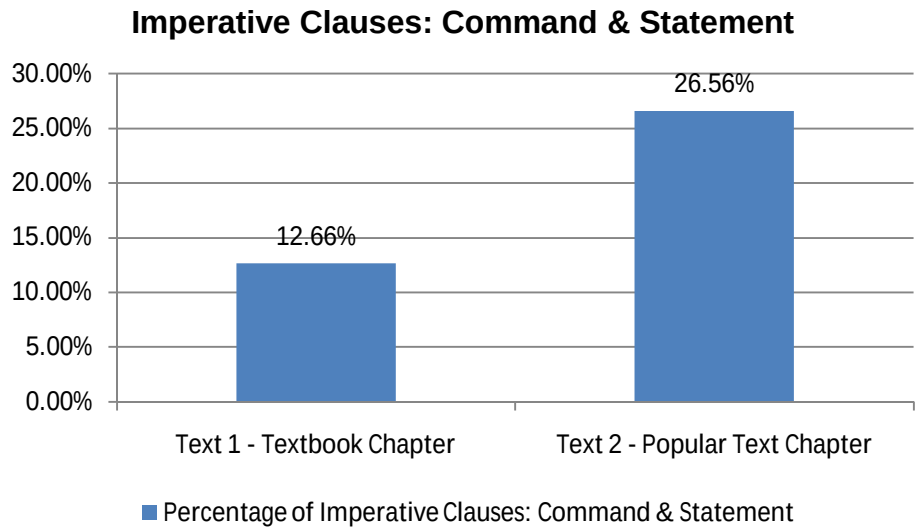

Figure 4.4: Distribution Patterns of Imperative Clauses: Command & Statement

It can be seen that Text 2 has 13.9% more Imperative clauses with Command and Statement than Text 1. This shows that Text 2 provides more advice in the form of an Imperative to encourage reader's participation in text. It can be concluded that Text 2 is more interactive and engages the reader's attention more than Text 1.

4.1.4 Distribution of the Metaphor of Mood

The distribution patterns of the Metaphor of Mood in Text 1 and Text 2 are shown in Table 4.13.

Table 4.13: Distribution of the Metaphor of Mood in Text 1 and Text 2

Text	Total No. of Sentences with the Metaphor of Mood	Total No. of Sentences in Text	Percentage	Calculation of Percentage
1	79	242	32.65%	(79/242) x 100%
2	64	173	36.99%	(64/173) x 100%

The distribution patterns of the Metaphor of Mood in Text 1 and Text 2 are illustrated in Figure 4.5.

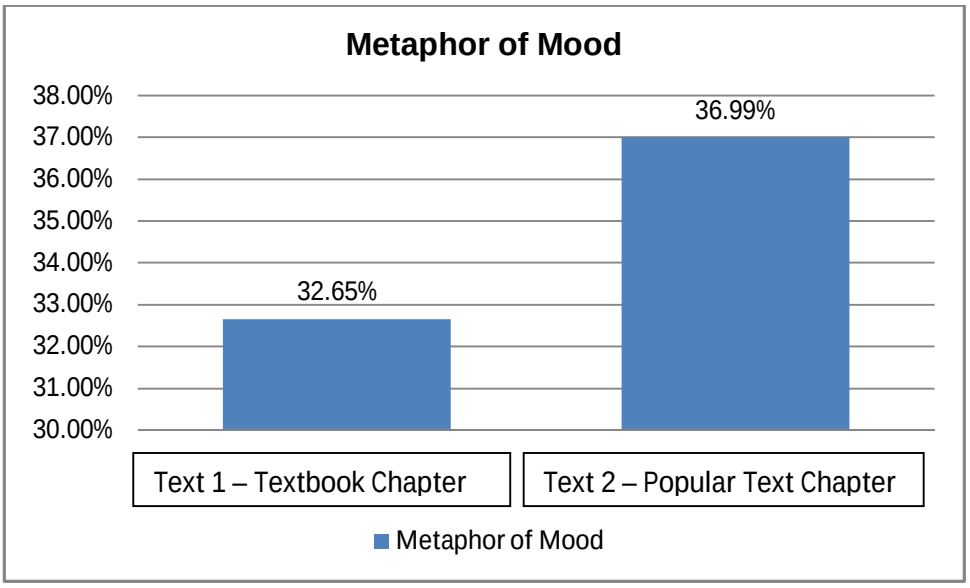

Figure 4.5: Distribution of Metaphor of Mood in Text 1 and Text 2

It can be seen that Text 2 has 4.34% more metaphorical clauses of Mood than Text 1. This shows that Text 2 is more interactive than Text 1, opening up more possibilities for semantic expansion within the metaphorical clause. Besides, with a higher number of metaphorical clauses, Text 2 appears to feature more lexicogrammatical features in the Mood meaning that contributes to the interactiveness of the text including politeness, giving advice and providing text direction for the text to be more accessible to readers. This is in line with the Dafouz-Milne's view that a higher occurrence of metaphorical clauses in a text makes the text more interactive and persuasive as discussed in Chapter 2. Although earlier analysis in Page 64 shows that Text 1 has more occurrences of metaphorical sentences than Text 2, the analysis involved only Declarative clauses. The conclusion that Text 2 is more interactive than Text 1 in Mood choices includes the occurrences of all three Mood types which are Declarative, Interrogative and Imperative.

4.2 Analysis of the Metaphor of Modality

As discussed in Chapter 3, the Metaphor of Modality is realized by the Mental and the Relational projection clauses that indicate 'probability' or 'obligation' (Halliday and Matthiessen, 2004).

4.2.1 Projection Clauses

The Metaphor of Modality is realized by two types of projection: Mental projections and Relational projections. Table 4.14 shows all instances of projection clauses in Text 1 and Table 4.15 shows all instances of projection clauses in Text 2.

Table 4.14: Projection Clauses in Text 1

Eg.	Label	Projection Clauses	Type of Projection
1	T1/INT/S2	**It would be tedious** || to have to write the following *statement* a hundred times.	Relational
2	T1/WL/S6	**It is always evaluated ||** be*fo*re the *loop* body is executed.	Relational
3	T1/FL/S26	**It is good programming practice** || to declare it in the *initial-action* of the *for loop*.	Relational
4	T1/MNE/S10	From this example, **you can see that** || a control variable can be a *float* type.	Mental
5	T1/MNE/S15	However, **you will be stunned to see that** || the result is actually *49.50000000000003.*	Mental
6	T1/MNE/S19	**The fundamental problem is that** || the *floating-point* numbers are represented by approximation.	Relational
7	T1/CS/S3	If you can write programs using *loops*, **you know** || how to program!	Mental
8	T1/FGCD/S6	**You know that** || number 1 is a common divisor, but it may not be the greatest common divisor.	Mental
9	T1/FGCD/S18	**It is important** || to think before you type.	Relational
10	T1/FGCD/S28	**You might think that** || a divisor for a number *n1* cannot be greater than *n1 / 2.*	Mental
11	T1/FSA/S10	**This suggests that** || you can try to find the *salesAmount* to match a given commission through incremental approximation.	Relational
12	T1/FSA/S14	This is a tedious job for humans, but **it is exactly what** || a computer is good for.	Relational
13	T1/KBC/S3	**It is generally used** || with an *if statement*.	Relational
14	T1/KBC/S6	**This keyword is generally used** || with an *if statement*.	Relational

Table 4.15: Projection Clauses in Text 2

Eg.	Label	Projection Clauses	Type of Projection
1	T2/LEO/S1	Have you ever been talking to someone and **it seems like** || he or she is saying the same thing over and over?	Relational
2	T2/LEO/S2	**I mean** ||, you keep listening, and they keep talking, and it all	Mental

		sounds the same.	
3	T2/LEO/S3	And they talk somemore and you listen somemore and **you wonder** || if it will ever end!	Mental
4	T2/LEO/S4	Congratulations, **you just experienced** || a perfect example of a verbal loop!	Mental
5	T2/GRL/S9	**You probably wonder** || exactly how the loop code works.	Mental
6	T2/GRL/S26	Whew, **that explanation seemed** || a little long-winded, and that's coming from the person that wrote it!	Relational
7	T2/GRL/S27	Unfortunately, **it isn't always easy** || to verbalize the flow of program code.	Relational
8	T2/GRL/S28	**This is why it's easy** || to fall back on figures.	Relational
9	T2/GRL/S30	**You can feel safe and secure knowing that** || I'm not running for president or trying to help you visualize my answer to global trade.	Mental
10	T2/GRL/S31	**I just want** || to help you learn how loops work!	Mental
11	T2/GRL/S41	**This is a very popular way** || to handle arrays.	Relational
12	T2/GRL/S42	**It is necessary ||** to subtract 1 in this case because all Java array indexes start with 0, which means they are zero based.	Relational
13	T2/GRL/S43	**It might be worth nothing that** || although zero-based arrays were used in other programming languages in the 1980s and before, they have nothing to do with the 80s movie Less than Zero or the 80s hit song saved by Zero.	Relational
14	T2/LJLW/S10	Because the while loop has no step expression, **it is important to make sure that** || the Statement somehow impacts the LoopCondition.	Relational
15	T2/LJLW/S11	Otherwise, **it is possible** || for the loop to repeat infinitely, which is usually a bad thing.	Relational
16	T2/LJLW/S18	**You can think of** || the while loop as a more general for loop.	Mental
17	T2/LJLW/S34	**The main concern is that** || the isCorrect () method returns a Boolean value that indicates whether or not the answer is correct.	Relational
18	T2/LJLW/S35	In this example, **it is impossible** || to know how many times the user will miss the answer and need the question repeated.	Relational
19	T2/TDNTD/S10	**It is always initially set** || during the first pass through the loop.	Relational
20	T2/TDNTD/S14	**This makes more sense** || than if it read "if the answer is not correct, ask the question and then check the answer again."	Relational
21	T2/AC/S2	**I used to** run across these pages and **wonder how** || you could make a page wait a few seconds and then automatically navigate to a new page.	Mental
22	T2/AC/S3	After I started programming in Java, **I realised** || what a trivial task this is.	Mental
23	T2/AC/S11	The main thing on which **I want you** || to focus is the page parameter, which is defined as:	Mental
24	T2/BA/S9	Of course, **it is rare that ||** you would purposely create an infinite loop and then use a break statement to bail out of it.	Relational
25	T2/LNK/S2	As dull as some humans can be, **I guarantee you** || computers are much duller when it comes to repeating the same thing over and over.	Mental

Section 4.2.1.1 shows the Mental projection clauses and Section 4.2.1.2 presents the Relational projection clauses.

4.2.1.1 Mental Projection Clauses

A Mental projection clause projects a clause with a Mental process verb that involves a person's cognition or feeling (Halliday, 1994).

(a) Identification of Mental Projection Clauses

Table 4.16 shows Mental Projection clauses in Text 1 and Table 4.17 shows Mental Projection clauses in Text 2.

Table 4.16: Mental Projection Clauses in Text 1

Eg.	Label	Sentence
1	T1/MNE/S10	From this example, **you can see that** || a control variable can be a *float* type.
2	T1/MNE/S15	However, **you will be stunned to see that** || the result is actually *49.50000000000003.*
3	T1/CS/S3	If you can write programs using *loops*, **you know** || how to program!
4	T1/FGCD/S6	**You know that** || number 1 is a common divisor, but it may not be the greatest common divisor.
5	T1/FGCD/S28	**You might think that** || a divisor for a number *n1* cannot be greater than *n1 / 2.*

Table 4.17: Mental Projection Clauses in Text 2

Eg.	Label	Mental Projection Clauses
1	T2/LEO/S2	**I mean** ||, you keep listening, and they keep talking, and it all sounds the same.
2	T2/LEO/S3	And they talk somemore and you listen somemore and **you wonder** || if it will ever end!
3	T2/LEO/S4	Congratulations, **you just experienced** || a perfect example of a verbal *loop*!
4	T2/GRL/S9	**You probably wonder exactly** || how the *loop* code works.
5	T2/GRL/S26	Whew, **that explanation seemed** || a little long-winded, and that's coming from the person that wrote it!
6	T2/GRL/S30	**You can feel safe and secure knowing that** || I'm not running for president or trying to help you visualize my answer to global trade.
7	T2/GRL/S31	**I just want** || to help you learn how *loops* work!
8	T2/LJLW/S18	**You can think** || of the *while loop* as a more general for *loop*.
9	T2/AC/S2	**I** used to run across these pages and **wonder how** || you could make a page wait a few seconds and then automatically navigate to a new page.
10	T2/AC/S3	After I started programming in *Java*, **I realised** || what a trivial task this is.
11	T2/AC/S11	The main thing on which **I want you** || to focus is the page parameter, which is defined as:
12	T2/LNK/S2	As dull as some humans can be, **I guarantee you** || computers are much duller when it comes to repeating the same thing over and over.

(b) Findings and Distribution of Mental Projection Clauses

Mental projection clauses have the lexicogrammatical features as discussed below. It should be noted that non-projection clauses are truncated due to limited space

For example: “**I just want ||** to help you learn how loops work!” Non-projection clause: “to help you learn how loops work!” is truncated.

b(i) Mental Verbs in Mental Projection Clauses

The verbs in Mental processes as shown in Table 4.16 and Table 4.17 are **see, know, experience, think, mean, wonder, feel, seem, want, realise** and **guarantee.** The Mental verbs represent processes of cognition, understanding and feeling as observed by Halliday (1994). The verb “see” is worth highlighting because it does not just involve the action of seeing, but it includes the readers understanding what they are seeing in the text. As for the verb “guarantee” (in this case, the author), it involves the author giving assurance and confidence to the reader.

b(ii) ‘Probability’ in Mental Projection Clauses with Modal Auxiliary Verbs

Modal auxiliary verbs indicating probability are also known as epistemic modal (Halliday, 1994). According to Halliday (1994), epistemic modals indicate a possibility of an event or idea. This is supported by Ravelli and Ellis (2004) who claim that epistemic modals are used to weaken claims and arguments. According to Ravelli and Ellis (2004:54), a writer construes an “authorial role, where his statement could be one possibility among the others”. This enables the readers to have alternative interpretations to address the issue being discussed.

The modal auxiliary verbs found in Mental projection clauses are **can, might** and **will.** Example 1 in Table 4.16, **you can see that…** has epistemic modal “can” that shows the possibility of the readers being able to understand the subject matter.

With epistemic modals, the clause appears friendlier and less imposing because the readers have the choice of disagreeing with the author's assumption. Besides, the epistemic modal also guides the reader towards the direction of the text explanation on the subject matter. Without this kind of guidance, the text will be more accessible to readers. Other similar clauses that share the same feature are in Examples 2 and 5 in Table 4.16 and in Example 6 in Table 4.17. It is to be noted that modal verb "will" has a higher probability value as compared to "can" and "might" (Halliday and Matthiessen, 2004).

b(iii) 'Probability' in Mental Projection Clauses without Modal Auxiliary Verbs

Mental projection clauses without modal auxiliary verbs are shown in Examples 3 and 4 in Table 4.16 and in Examples 2, 3, 4, 5 and 9 in Table 4.17. The clauses are based on the author's assumption indicating probability in events such as **you wonder …** and **you know that…**

These projection clauses also state facts that indicate milestones in the direction of the text such as in Example 4 in Table 4.17, "**you probably wonder exactly ||** how the *loop* code works". Besides, these projection clauses also serve as reminders to readers like in Example 4 in Table 4.16, "**you know that ||** number 1 is a common divisor…"

b(iv) 'Obligation' in Mental Projection Clauses with Modal Auxiliary Verbs

Modal Auxiliary verbs indicating 'obligation' are also known as deontic modals (Halliday and Matthiessen, 2004). According to Halliday (1994), deontic modals indicate a necessity of action. This is in line with Ravelli and Ellis's (2004)

discussion on deontic modals that are used by writers to give advice to the readers. As described by Ravelli and Ellis (2004), writers construe a "consultant role" in telling what the readers should do.

To illustrate, Example 8 in Table 4.17, **you can think…** indicates an action of "thinking" to be done by the reader. It also functions as an advice for the reader to understand the subject matter easily.

b(v) 'Obligation' in Mental Projection Clauses without Modal Auxiliary Verbs

It is found that all 'obligation' clauses without modal auxiliary verbs indicate the writer's involvement in the text. In this case, the action is to be carried out by the writer, instead of the reader such as in Example 7 in Table 4.17, **I just want …** Examples 1, 10, 11 and 12 in Table 4.17 share the similar feature.

b(vi) Grammatical Persons in Mental Projection Clauses

Two types of grammatical persons are found in Tables 4.16 and 4.17. It is found that Text 1 has no occurrences of "I" and 5 occurrences of "you". Text 2 shows 6 occurrences of "I" and 7 occurrences of "you".

While Text 1 has no indication of the author's involvement, the writer of Text 2 appears to place himself in the forefront of the text. This is to enhance the interactivity and solidify the relationship between the reader and the writer.

b (vii) Modal Adjuncts in Mental Projection Clauses

Modal Adjuncts are grammatical constituents that contribute to the interpersonal meaning and influence the interactiveness of the text. This is done through "adding an expression of attitude" in the text (Eggins, 2004:160).

It is found that Text 2 has three modal adjuncts but Text 1 has none.

- **just** (Examples 3 and 7 in Table 4.17) adds the writer's expression and denotes a casual tone to the text.
- **exactly** (Example 4 in Table 4.17) indicates a higher accuracy of the situation.
- **probably** (Example 4 in Table 4.17) indicates the probability of the situation.

(c) Distribution of Mental Projection Clauses

It is found that Text 1 has 5 occurrences of Mental Projection clauses and Text 2 has 12 occurrences. Table 4.18 shows the distribution of Mental Projection Clauses in Text 1 and Text 2 in percentage.

Table 4.18: Distribution of Mental Projection Clauses

Text	No. of Mental Projection Clauses	Total No. of Sentences with the Metaphor of Modality	Percentage of Mental Projection Clauses	Calculation of Percentages
1	5	14	35.71%	(5/14) x 100%
2	12	26	46.15%	(12/26) x 100%

Figure 4.6 illustrates the distribution of Mental Projection in Text 1 and Text 2.

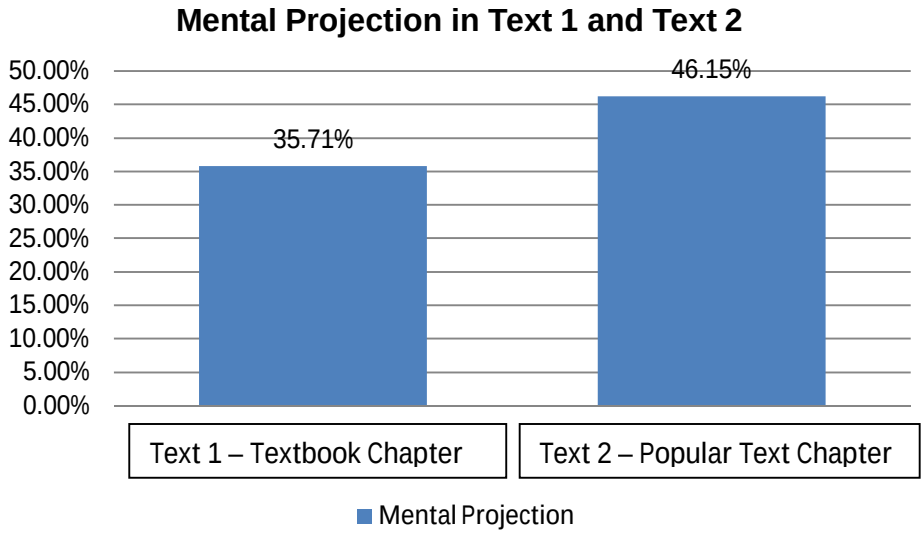

Figure 4.6: Mental Projection in Text 1 and Text 2

Text 2 has 10.44% more Mental Projections that Text 1. This means that Text 2 features more of the author's personal assessment towards the subject issue. With a higher percentage of Mental Projection, Text 2 appears to be friendlier than Text 1 since the author of Text 2 positions himself in the forefront of the text, making the text arguable and interactive.

Table 4.19 shows the distribution of the Mental projection types 'probability' and 'obligation'.

Table 4.19: Distribution of Mental Projection Types in Text 1 and Text 2

Text	Total No. of Mental Projection Clauses	Type of Mental Projection			
		Probability	Calculation of Percentage	Obligation	Calculation of Percentage
1	5	4 (80.00%)	(4/5) x 100%	1 (20.00%)	(1/5) x 100%
2	12	7 (58.33%)	(7/12) x 100%	5 (41.67%)	(5/12) x 100%

Figure 4.7 illustrates the distribution of Mental projection types in Text 1.

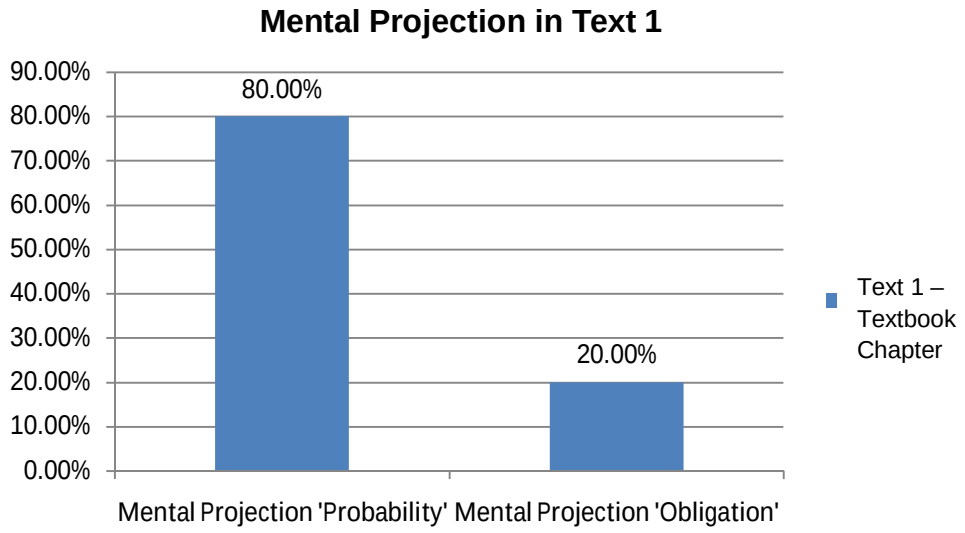

Figure 4.7: Mental Projection Types in Text 1

It can be seen that Text 1 has 60% more ‘probability’ Mental Projection clauses than ‘obligation’. This means that the author of Text 1 addresses the subject matter with more prediction on probability of events instead of demanding for actions to be carried out by the readers.

Figure 4.8 illustrates the distribution of the Mental projection types in Text 2.

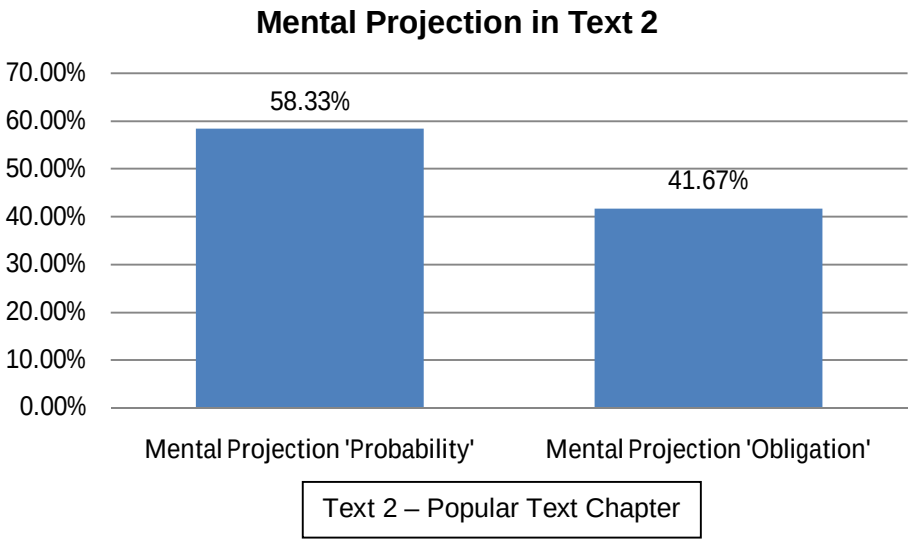

Figure 4.8: Mental Projection Types in Text 2

Similar to the Mental Projection distribution patterns in Text 1, Text 2 has a higher percentage of ‘probability’ type of Mental Projection compared to the ‘obligation’ type. This also means that the author includes his personal judgment based on the possibility of events such as predicting the reader’s thoughts and conclusions to make the text friendlier.

4.2.1.2 Relational Projection Clauses

The Relational projection clauses have the processes of being and having (Halliday, 1994).

(a) Identification of Relational Projection Clauses

Table 4.20 shows Relational Projection clauses in Text 1 and Table 4.21 shows Relational Projection clauses in Text 2.

Table 4.20: Relational Projection Clauses in Text 1

Eg.	Label	Relational Projection Clauses
1	T1/INT/S2	**It would be tedious** || to have to write the following *statement* a hundred times.
2	T1/WL/S6	**It is always evaluated ||** before the *loop* body is executed.
3	T1/FL/S26	**It is good programming practice** || to declare it in the *initial-action* of the *for loop*.
4	T1/MNE/S19	**The fundamental problem is that** || the *floating-point* numbers are represented by approximation.
5	T1/FGCD/S18	**It is important** || to think before you type.
6	T1/FSA/S10	**This suggests that** || you can try to find the *salesAmount* to match a given commission through incremental approximation.
7	T1/FSA/S14	This is a tedious job for humans, but **it is exactly** || what a computer is good for.
8	T1/KBC/S3	**It is generally used** || with an *if statement*.
9	T1/KBC/S6	**This keyword is generally used** || with an *if statement*.

Table 4.21: Relational Projection Clauses in Text 2

Eg	Label	Relational Projection Clauses
1	T2/LEO/S1	Have you ever been talking to someone and **it seems like** || he or she is saying the same thing over and over?
2	T2/GRL/S27	Unfortunately, **it isn't always easy** || to verbalize the flow of program code.
3	T2/GRL/S28	**This is why it's easy** || to fall back on figures.
4	T2/GRL/S41	**This is a very popular way** || to handle arrays.
5	T2/GRL/S42	**It is necessary ||** to subtract 1 in this case because all Java array indexes start with 0, which means they are zero based.
6	T2/GRL/S43	**It might be worth noting that** || although zero-based arrays were used in other programming languages in the 1980s and before, they have nothing to do with the 80s movie Less than Zero or the 80s hit song saved by Zero.
7	T2/LJLW/S10	Because the while *loop* has no step expression, **it is important to make sure that** || the *Statement* somehow impacts the LoopCondition.
8	T2/LJLW/S11	Otherwise, **it is possible** || for the *loop* to repeat infinitely, which is usually a bad thing.
9	T2/LJLW/S34	**The main concern is that** || the *isCorrect () method* returns a Boolean value that indicates whether or not the answer is correct.
10	T2/LJLW/S35	In this example, **it is impossible** || to know how many times the user will miss the answer and need the question repeated.
11	T2/TDNTD/S10	**It is always initially set** || during the first pass through the *loop*.
12	T2/TDNTD/S14	**This makes more sense** || than if it read "if the answer is not correct, ask the question and then check the answer again."
13	T2/BA/S9	Of course, **it is rare that ||** you would purposely create an infinite *loop* and then use a *break statement* to bail out of it.

(b) Findings and Distribution of Relational Projection Clauses

It is found that Relational Projection clauses have the lexicogrammatical features as discussed below. The non projection clauses are truncated due to limited space.

For example: **This is why it's easy ||** to fall back on figures. The non-projection clause "to fall back on figures" is truncated.

b(i) 'Probability' in Relational Projection Clauses – Epistemic Modality

Clauses that denote 'probability' are in Examples 1, 3, 4, 5, 6 and 7 in Table 4.20 and in Examples 1, 2, 3, 4, 5, 9, 10, 11 and 13 in Table 4.21. Example 1 in Table 4.20, **It would be tedious to have to write the following *statement* a hundred times** shows that there is a possibility that readers may find "writing the following *statement* a hundred times" as a "tedious" action.

b(ii) 'Obligation' in Relational Projection Clauses – Deontic Modality

Clauses that denote 'obligation' are in Examples 2, 8 and 9 in Table 4.20 and in Examples 6, 7, 8 and 12 in Table 4.21. Example 7 in Table 4.21, **it is important to make sure that…** denotes the reader as having the responsibility of carrying out the action of "making sure". Another clause is in Example 2 in Table 4.20, **It is always evaluated …** This obligation for an action to be carried out in this clause is neither the reader nor the writer, but the computer programme itself. The writer refers to the computer programme as one of the stakeholders in the text.

b(iii) Absence of Grammatical Persons

No grammatical persons are found in Table 4.20 and Table 4.21. Relational projection clauses display a high level of truth in the statements of the author and hence the removal of agency renders the text non-arguable and non-negotiable

(Ravelli & Ellis, 2004). Text 2's writer presents more concretised facts to the readers compared to Text 1.

b(iv) Attributes in Relational Projection Clauses

It is found that some Relational projection clauses have attributes to denote the author's opinion or attitude about the subject matter. Example 1 in Table 4.20, **It would be tedious to …** has an attribute "tedious" to "signal the speaker's position" and to include this personal judgment on the subject matter (Stillar, 1998:36). The author uses attributes to make the text more persuasive to readers.

b(v) Modal Adjuncts in Relational Projection Clauses

Modal Adjuncts are used to orientate the readers towards the speaker's attitudes towards the text without involving the speaker in the foreground of the text. Hence, the speaker's attitudes are non-arguable in the text. Some modal adjuncts found in Tables 4.20 and 4.21 are **always, exactly, generally, possible, and impossible.**

(c) Distribution of Relational Projection Clauses

The distribution of Relational Projection Clauses is shown in Table 4.22.

Table 4.22: Distribution of Relational Projection Clauses

Text	No. of Relational Projection Clauses	Total No. of Sentences with the Metaphor of Modality	Percentage	Calculation of Percentages
1	9	14	64.29%	(9/14) x 100%
2	14	26	56.00%	(14/26) x 100%

Figure 4.9 illustrates the distribution of Relational Projection clauses in Text 1 and Text 2.

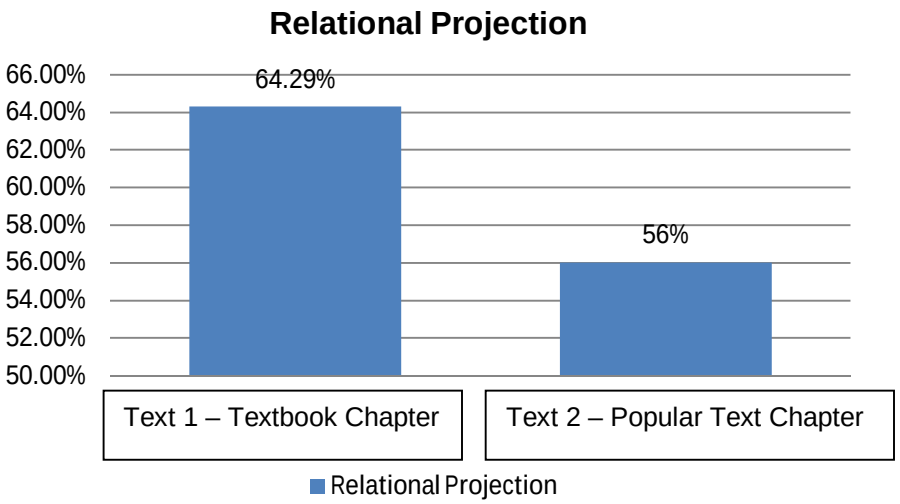

Figure 4.9: Relational Projection in Text 1 and Text 2

Text 1 has a higher percentage of 8.29% than Text 2's Relational Projection in the text. This shows that Text 1 has more author's objective claims of certainty. This also means that Text 1 has more inarguable clauses compared to Text 2. Readers of Text 1 have no alternatives but to accept the author's claim as it is. Text 2 has a lower percentage of Relational Projection where readers have the choice to accept or reject the author's statement in the text. Hence, Text 2 is more interactive than Text 1.

Table 4.23 shows the distribution of Relational Projection clauses based on the types of 'probability' and 'obligation'.

Table 4.23: Distribution of Relational Projection Clauses in Text 1 and Text 2

Text	No of Relational Projection Clauses	Type of Relational Projection			
		Probability	Calculation of Percentage	Obligation	Calculation of Percentage
1	9	6 (66.67%)	(6/9) x 100%	3 (33.33%)	(3/9) x 100%
2	14	11 (78.57%)	(11/14) x 100%	3 (21.43%)	(3/14) x 100%

The distribution of Relational Projection clauses in Text 1 is shown in Figure 4.10.

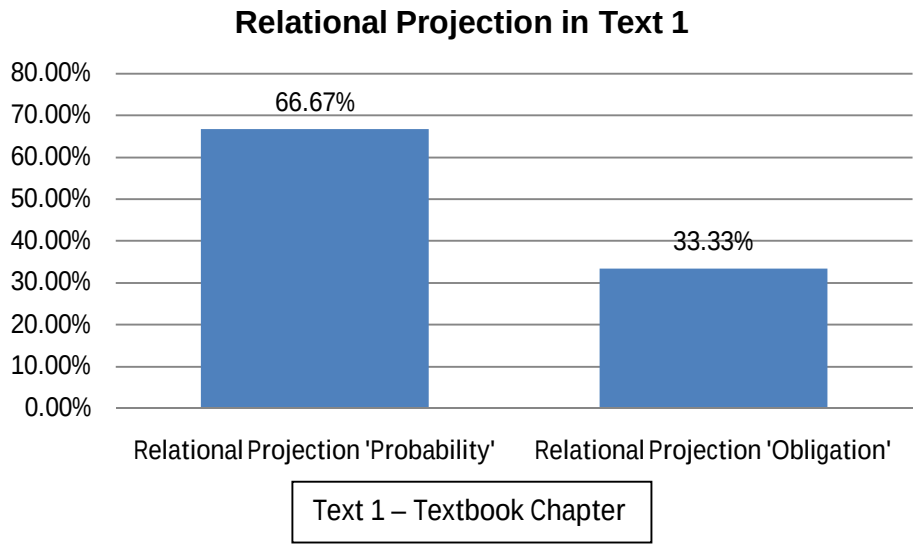

Figure 4.10: Distribution of Relational Projection Clauses in Text 1

It can be seen that Text 1 has more ‘probability’ type of Relational Projection clauses that features the author’s high value of certainty in the text without involving himself in the text. This makes the text non-arguable since in Relational Projection, there is no mention of any persons involved in making those claims, hence the claims are concretized. Therefore, Text 1 is a less interactive and less persuasive text since the readers have no options to argue or have alternative ideas towards the text.

The distribution of Relational Projection clauses in Text 2 is shown in Figure 4.11.

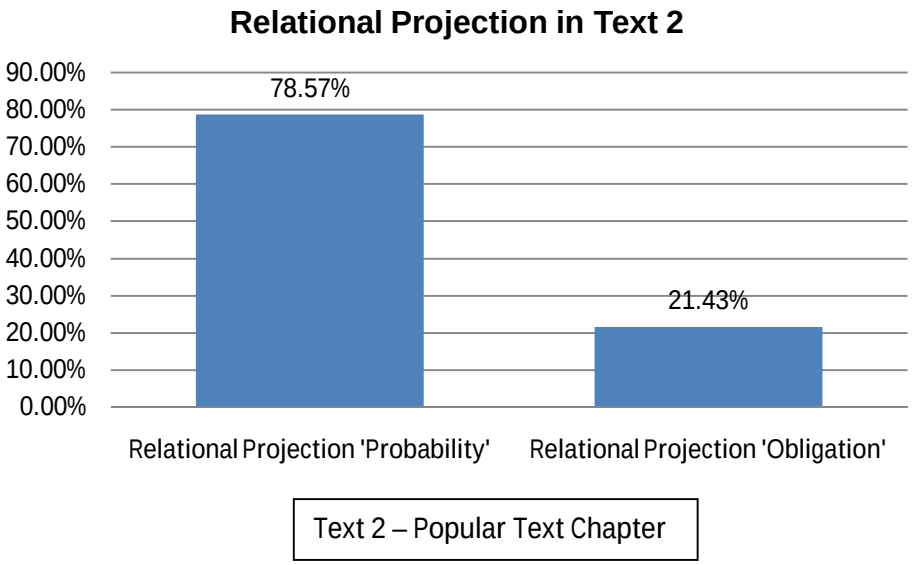

Figure 4.11: Distribution of Relational Projection Clauses in Text 2

Text 2 shows a similar pattern to Text 1 in the distribution of Relational Projection types. Text 2 has a higher percentage of ‘Probability’ Relational Projection. This shows that Text 2’s author attempts to make the text more persuasive instead of requesting for a form of action to be carried out by the reader. This feature leads to a more persuasive text.

4.2.2 Distribution of Projection Clauses as Metaphor of Modality

The distribution of projection clauses as Metaphor of Modality is shown in Table 4.24.

Table 4.24: Distribution patterns of Metaphor of Modality in Text 1 and Text 2

Text	No. of Sentences with Metaphor of Modality	Total no. of Sentences in Text	Percentage	Calculation of Percentage
Text 1	14	242	5.78%	(14/242) x 100%
Text 2	26	173	15.03%	(25/173) x 100%

The distribution of projection clauses as Metaphor of Modality is illustrated in Figure 4.12.

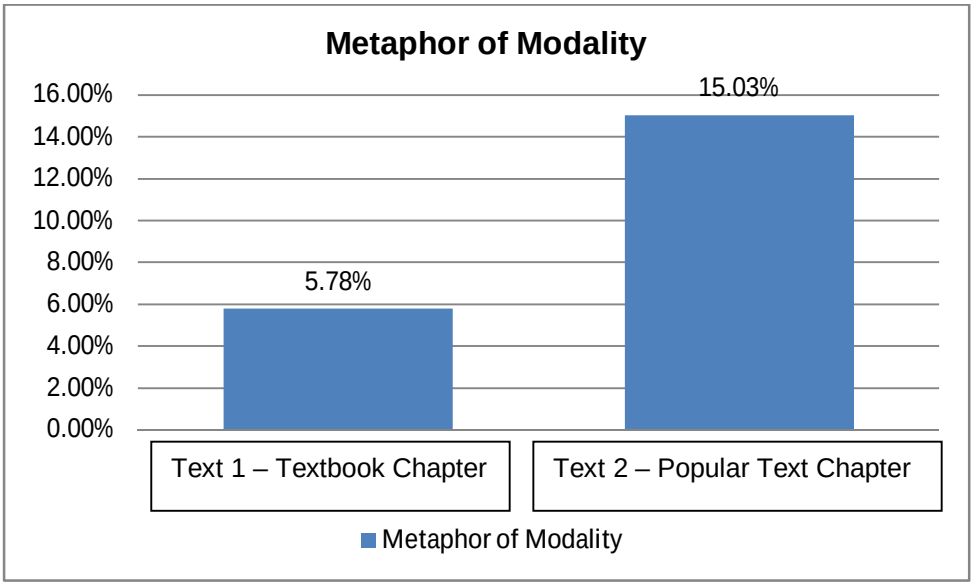

Figure 4.12: Distribution Patterns of Metaphor of Modality in Text 1 and Text 2

Figure 4.12 shows that both texts show an evidence of persuasion. Text 2 has 9.25% more Metaphor of Modality in comparison to Text 1. This shows that Text 2 is more metaphorical in text. Text 2 includes more of the author's personal judgement through

Mental projection and having high value of claims to make the text more accessible, persuasive and interactive.

4.3 Distribution of Interpersonal Metaphor

The distribution of Interpersonal Metaphor is shown in Table 4.25.

Table 4.25: Distribution of Interpersonal Metaphor

Text	Total No. of Sentences in Text	No. of Sentences with the Metaphor of Mood	Calculation of Percentage	No. of Sentences with the Metaphor of Modality	Calculation of Percentage
1	242	79 (32.64%)	(79/242) x 100%	14 (5.79%)	(14/242) x 100%
2	173	64 (36.99%)	(64/173) x 100%	26 (15.03%)	(26/173) x 100%

The distribution of Interpersonal Metaphor is illustrated in Figure 4.13.

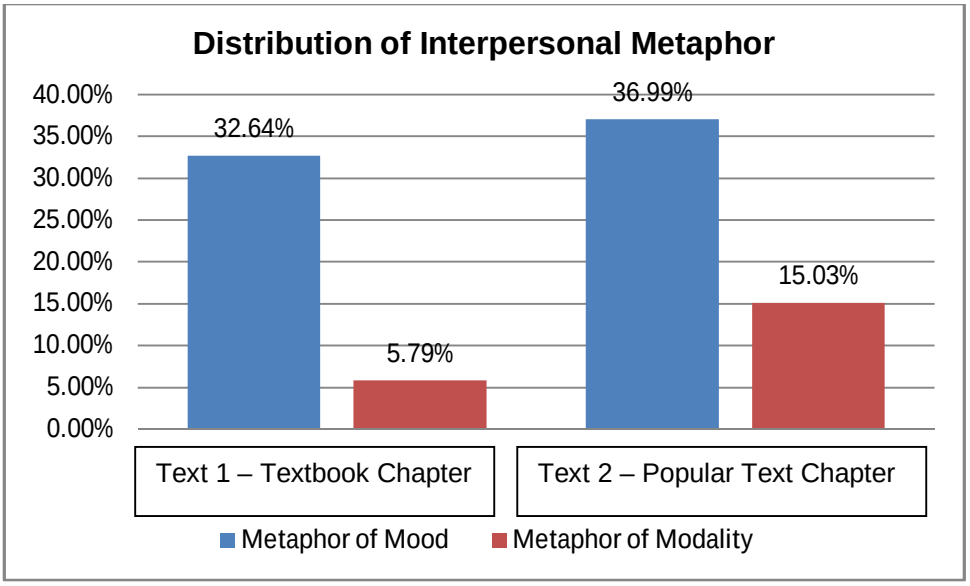

Figure 4.13: Distribution of Interpersonal Metaphor in Text 1 and Text 2

As shown in Figure 4.13, it can be seen that Text 2 is more metaphorical than Text 1 in the use of Metaphor of Mood and Metaphor of Modality. Text 2 has 4.35% more Metaphor of Mood than Text 1. This shows that Text 2 has more potential for semantic expansions compared to Text 1. Besides, Text 2 has 9.24% more Metaphor of

Modality than Text 1. This shows that Text 2 has more projection clauses enabling the inclusion of the author's personal judgment in the text. The author's position is strengthened, making the text more reliable since the author foregrounds his position in the text to make the text arguable. Thus it can be said that Text 2 is more persuasive, accessible and interactive for readers than Text 1. This is because Text 2 has more metaphorical clauses of Mood and Modality compared to Text 1.

4.4 Chapter Summary

This chapter has presented the analysis and findings of the study, including the distribution patterns of Metaphor of Mood and Metaphor of Modality. Chapter 5 concludes the study.