10 References

[1] M. P. Bhopal, "Natural language Interface for Database: A Brief review," *IJCSI*, p. 600, 2011.

[2] B. Hemerelain and H. Belbachir, "Semantic Analysis of Natural Language Queries for an Object Oriented Database," *J. Softw. Eng. Appl.*, vol. 3, no. 11, pp. 1047–1053, 2010.

[3] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases–an introduction," *Nat. Lang. Eng.*, vol. 1, no. 1, pp. 29–81, 1995.

[4] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," in *Proceedings of the 8th international conference on Intelligent user interfaces*, 2003, pp. 149–157.

[5] "Patent US5495604 - Method and apparatus for the modeling and query of database structures using ... - Google Patents." [Online]. Available: https://www.google.com/patents/US5495604. [Accessed: 16-Dec-2016].

[6] "Academic paper: The Lunar Science Natural Language Information System: Final Report." [Online]. Available: https://www.researchgate.net/publication/247926251_The_Lunar_Science_Natural_Language_Information_System_Final_Report. [Accessed: 29-Apr-2017].

[7] D. Waltz and B. Goodman, "Planes: A Data Base Question-answering System," *SIGART Bull*, no. 61, pp. 24–24, Feb. 1977.

[8] G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a natural language interface to complex data," *ACM Trans. Database Syst. TODS*, vol. 3, no. 2, pp. 105–147, 1978.

[9] R. Alexander, P. Rukshan, and S. Mahesan, "Natural Language Web Interface for Database (NLWIDB)," *ArXiv Prepr. ArXiv13083830*, 2013.

[10] B. B. Huang, G. Zhang, and P. C. Y. Sheu, “A Natural Language Database Interface Based on a Probabilistic Context Free Grammar,” in *IEEE International Workshop on Semantic Computing and Systems*, 2008, pp. 155–162.

[11] M. J. Minock, “A STEP towards realizing Codd’s vision of rendezvous with the casual user,” in *Proceedings of the 33rd international conference on Very large data bases*, 2007, pp. 1358–1361.

[12] E. F. Codd, *Seven Steps to Rendezvous with the Casual User*. IBM Corporation, 1974.

[13] D. H. D. Warren and F. C. N. Pereira, “An Efficient Easily Adaptable System for Interpreting Natural Language Queries,” *Comput Linguist*, vol. 8, no. 3–4, pp. 110–122, Jul. 1982.

[14] D. L. Waltz, “An English language question answering system for a large relational database,” *Commun. ACM*, vol. 21, no. 7, pp. 526–539, Jul. 1978.

[15] B. H. Thompson and F. B. Thompson, “Introducing ask, a simple knowledgeable system,” presented at the Proceedings of the first conference on Applied natural language processing, 1983, pp. 17–24.

[16] P. Resnik, “Access to Multiple Underlying Systems in Janus,” BBN SYSTEMS AND TECHNOLOGIES CORP CAMBRIDGE MA, BBN-7142, Sep. 1989.

[17] M. Templeton and J. Burger, “Problems in Natural-language Interface to DBMS with Examples from EUFID,” in *Proceedings of the First Conference on Applied Natural Language Processing*, Stroudsburg, PA, USA, 1983, pp. 3–16.

[18] C. D. Hafner, “Interaction of Knowledge Sources in a Portable Natural Language Interface,” in *Proceedings of the 10th International Conference on Computational Linguistics and 22Nd Annual Meeting on Association for Computational Linguistics*, Stroudsburg, PA, USA, 1984, pp. 57–60.

[19] B. J. Grosz, “TEAM: a transportable natural-language interface system,” presented at the Proceedings of the first conference on Applied natural language processing, 1983, pp. 39–45.

[20] C. Reviewer-Lee, “Book review: Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory by Risto Miikkulainen (Bradford Books, MIT Press 1993),” *ACM SIGART Bull.*, vol. 6, no. 4, pp. 19–21, Oct. 1995.

[21] E. Charniak, *Statistical Language Learning*. MIT Press, 1994.

[22] K. W. Church and R. L. Mercer, “Introduction to the special issue on computational linguistics using large corpora,” *Comput. Linguist.*, vol. 19, no. 1, pp. 1–24, Mar. 1993.

[23] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of English: the penn treebank,” *Comput. Linguist.*, vol. 19, no. 2, pp. 313–330, Jun. 1993.

[24] R. G. Reilly, Ed., *Connectionist approaches to natural language processing*. Hove: Lawrence Erlbaum Assoc, 1992.

[25] L. Shastri, “A model of rapid memory formation in the hippocampal system,” in *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, 1997, pp. 680–685.

APPENDIX A – Class Program – NLIDB Converter

```
using edu.stanford.nlp.ling;
using edu.stanford.nlp.pipeline;
using edu.stanford.nlp.util;
using java.io;
using java.util;
using System;
using System.Collections.Generic;
using System.IO;
using Console = System.Console;

namespace NLIDBConsoleApplication {
 internal class Program {
  public static void Main(string[] args) {
   NLIDBLogger nlidbl = new NLIDBLogger();
   NLIDBBAL nlidbbal = new NLIDBBAL();

   List < string > nnTokens = new List < string > ();

   int ruleIdNnt = 0;
   int ruleIdNnv = 0;
   int ruleIdNnc = 0;

   string dependencies = null;

   int tokenId = 0;
   int nnTokenId = 0;
   int parentTokenId = 0;
   byte tokenTypeId = 0;
   int dependencyId = 0;
   int sentencesCount = 0;

   // PATH TO THE FOLDER WITH MODELS EXTRACTED FROM `stanford-corenlp-3.7.0-models.jar`
   var jarRoot = @ "H:\MSCIT-14-069\RESEARCH\NLIDBConsoleApplication\stanford-corenlp-
full-2016-10-31\stanford-corenlp-3.7.0-models";

   Console.WriteLine("\n  NLIDB Converter for the Customer Relationship Index starting
up...\n");

   // TEXT FOR PROCESSING
   Console.WriteLine("  Type your question in Natural Language:\n");

   var question = Console.ReadLine().Trim();

   Console.WriteLine();

   // CHECKS IF THE INPUT STRING IS NOT EMPTY
   if (question != null && question.Length > 1) {
    nlidbl.CreateLogHeader(question);
    Console.WriteLine(nlidbl.MessageCheckingKBForExistingQuestion);
    nlidbl.PrintMessage(nlidbl.MessageCheckingKBForExistingQuestion);

    string userName = Environment.UserName;

    // CHECK IF THE QUESTION ALREADY EXIST IN THE QUESTIONS KNOWLEDGE BASE
    if (nlidbbal.IsQuestionInKB(question)) {
     // IF QUESTION IS IN KB
     Console.WriteLine();
     Console.WriteLine(nlidbl.MessageExistingQuestion);
     nlidbl.PrintMessage(nlidbl.MessageExistingQuestion);

     // GETS THE CORRESPONDING QUESTION ID BASED ON THE INPUT STRING
     int questionId = nlidbbal.GetIdOfNaturalLanguageQuestion(question);

     // GETS THE SQL STATEMENT BASED ON THE QUESTION ID
     string sqlStatement = nlidbbal.GetSqlStatementForQuestionID(questionId);

     // IF A VALID SQL STATEMENT IS RETURNED
     if (sqlStatement.Substring(0, 6).ToLower() == "select") {
      Console.WriteLine();
      Console.WriteLine(sqlStatement);
      nlidbl.PrintMessage(sqlStatement);
     } else // IF THE NATURAL LANGUAGE QUESTION WAS NOT CONVERTED TO SQL
SUCCESSFULLY
     {
      Console.WriteLine();
      Console.WriteLine(sqlStatement);
      nlidbl.PrintMessage(sqlStatement);
      nlidbl.PrintMessage(nlidbl.MessageSQlConversionFailed);
     }
     Console.ReadLine();
    } else // IF QUESTION IS NOT IN KB
    {
     Console.WriteLine();
     Console.WriteLine(nlidbl.MessageNewQuestion);
     nlidbl.PrintMessage(nlidbl.MessageNewQuestion);

     // ADD THE QUESTION IN TO THE QUESTIONS KB
     int questionId = nlidbbal.AddNewQuestionToKB(question, userName);

     Console.WriteLine();
     Console.WriteLine(nlidbl.MessageNlidbConverterExecutionStart);
     nlidbl.PrintMessage(nlidbl.MessageNlidbConverterExecutionStart);

     // STANFORD CORE NLP ANNOTATION PIPELINE CONFIGURATION
     var props = new Properties();
     props.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse,
dcoref");
     props.setProperty("ner.useSUTime", "0");

     // CHANGE CURRENT DIRECTORY, SO STANFORD CORENLP COULD FIND ALL     THE MODEL
FILES AUTOMATICALLY
     var curDir = Environment.CurrentDirectory;
     Directory.SetCurrentDirectory(jarRoot);
     var pipeline = new StanfordCoreNLP(props);
     Directory.SetCurrentDirectory(curDir);

     // ANNOTATION
     var annotation = new Annotation(question);
     pipeline.annotate(annotation);

     Console.WriteLine();
     Console.WriteLine(nlidbl.MessageOutputFromStanfordCoreNlp);
     nlidbl.PrintMessage(nlidbl.MessageOutputFromStanfordCoreNlp);

     // RESULT - PRETTY PRINT
     using(var stream = new ByteArrayOutputStream()) {
      pipeline.prettyPrint(annotation, new PrintWriter(stream));
      Console.WriteLine();
      Console.WriteLine(stream.toString());
```

```
       nlidbl.PrintMessage(stream.toString());
       stream.close();
       Console.WriteLine("================================");
      }

      //EXTRACTING SENTENCES FROM THE INPUT STRRING
      ArrayList sentences = annotation.get(new
CoreAnnotations.SentencesAnnotation().getClass()) as ArrayList;

      // GETTING THE COUNT OF SENTENSES IN THE INPUT TEXT
      sentencesCount = sentences.size();

      // NLIDB CONVERTER RUNS IF THE SENTENSES COUNT IS = MAX_SENTENSES,
THIS DOES NOT SUPPORT MULTIPLE SENTENSES
      if (sentencesCount == nlidbbal.MaxSentenses) {
       foreach(CoreMap sentence in sentences) {
        Console.WriteLine();
        Console.WriteLine(nlidbl.MessagePrintingSentense);
        nlidbl.PrintMessage(nlidbl.MessagePrintingSentense);

        Console.WriteLine();
        Console.WriteLine(sentence.ToString());
        nlidbl.PrintMessage(sentence.ToString());

        //EXTRACTING TOKENS IN THE SENTENCE
        var tokens = sentence.get(new CoreAnnotations.TokensAnnotation().getClass()) as
ArrayList;

        Console.WriteLine();
        Console.WriteLine(nlidbl.MessagePrintingTokens);
        nlidbl.PrintMessage(nlidbl.MessagePrintingTokens);

        int tokensCount = 1;

        //EXTRACT THE PROPERTIES OF EACH TOKEN IN TOKENS
        foreach(CoreLabel token in tokens) {
         // GET THE TOKEN_TEXT
         string tokenText = token.get(new
CoreAnnotations.TextAnnotation().getClass()).ToString();
         // GET THE PART OF SPEECH TAG OF TOKEN BASED ON THE PENN TREE BANK PART OF
SPEECH TAGS
         string partOfSpeechTag = token.get(new
CoreAnnotations.PartOfSpeechAnnotation().getClass()).ToString();

         // GET THE TOKEN TYPE ID BASED ON THE INPUT PART OF SPEECH TAG
         tokenTypeId = nlidbbal.TokenTypeIDForInputString(partOfSpeechTag);
         // SAVE THE TOKEN TO THE DATABASE AGAINST THE QUESTION ID AND TOKEN TYPE ID
         tokenId = nlidbbal.AddNewTokenToQuestion(tokenText, questionId, tokenTypeId);

         Console.WriteLine();
         Console.WriteLine("T :" + tokensCount + " || Token :" + tokenText + " || Part
of Speech Tag :" + partOfSpeechTag);
         nlidbl.PrintMessage("T :" + tokensCount, tokenText, partOfSpeechTag);

         tokensCount = tokensCount + 1;
        }

        // EXTRACT TOKEN DEPENDENCIES IN THE SENTENSE USING STANFORD PARSER
        dependencies = nlidbbal.GetTokenDependencies(sentence.ToString());
        // SPLITTING THE DEPENDENCIES STRING TO EXTRACT THE TOKEN AND PARENT TOKEN -
 DEFINING SEPERATORS
        string[] stringSeparators = new string[] {
         "),"
        };
        // SPLITTING THE DEPENDENCIES STRING TO EXTRACT THE TOKEN AND PARENT TOKEN
        var dependencyList = dependencies.Split(stringSeparators,
 StringSplitOptions.None);

        // EXTRACT THE TOKEN AND PARENT TOKEN OF EACH DEPENDENCY IN DEPENDENCY_LIST
        foreach(string dependency in dependencyList) {
         // SPLITTING THE DEPENDENCY TO EXTRACT THE TOKEN AND PARENT TOKEN - DEFINING
 INNER SEPERATORS
         string[] innerStringSeperators = new string[] {
          "(",
          ",",
          "-"
         };
         // SPLITTING THE DEPENDENCIES STRING TO EXTRACT THE TOKEN AND PARENT TOKEN
         var tokenDependencies = dependency.Split(innerStringSeperators,
 StringSplitOptions.None);

         if (tokenDependencies[1] == "ROOT") {
          parentTokenId = 0;
          // GET THE TOKEN ID FOR CHILD TOKEN
          tokenId = nlidbbal.GetMatchingTokenId(questionId, tokenDependencies[3]);
          int X = nlidbbal.GetTokenIdInQuestion(questionId, tokenDependencies[3]);
         } else {
          // GET THE TOKEN ID FOR PARENT TOKEN
          parentTokenId = nlidbbal.GetMatchingTokenId(questionId, tokenDependencies[1]);
          // GET THE TOKEN ID FOR CHILD TOKEN
          tokenId = nlidbbal.GetMatchingTokenId(questionId, tokenDependencies[3]);
         }

         // SAVE THE PARENT AND CHILD TOKEN TO THE DATABASE
         dependencyId = nlidbbal.AddTokenDependencies(tokenId, parentTokenId,
 tokenDependencies[0]);
        }

        // GET THE NN TOKENS BASED ON THE PENN TREE BANK TAGS AND MATCH THEM WITH SQL
 NODE TYPES
        // GET A LIST OF NN TOKENS FOR QUESTION
        nnTokens = nlidbbal.GetPennTreeBankPartOfSpeechTagsForQuestion_NN(questionId);

        // EACH TOKEN IS CHECKED AGAINS THE NATURAL LANGUAGE RULES FOR SQL NODE TYPES
 NNT(NAME NODE TABLE) AND NNV(NAME NODE VIEW)
        foreach(var nnToken in nnTokens) {
         // GET THE NATURAL LANGUAGE RULE ID FOR NAME NODE TABLE
         ruleIdNnt = nlidbbal.GetNaturalLanguageRulesForSqlNodeTypeNnt(nnToken);

         if (ruleIdNnt != 0) // IF A VALID RULE EXIST
         {
          // GET THE TOKEN ID
          nnTokenId = nlidbbal.GetTokenIdInQuestion(questionId, nnToken);
          // SAVE TOKEN AND NATURAL LANGUAGE RULE TO DATABASE - THE NATURAL LANGUGAE
 RULE IS MAPPED TO THE SQL NODE TYPE. THEREFORE THE TOKEN IS MAPPED TO THE SQL NODE NNT
          nlidbbal.AddToTokensMappedToSqlNode(nnTokenId, ruleIdNnt);
         }

         // GET THE NATURAL LANGUAGE RULE FOR NAME NODE VIEW
         ruleIdNnv = nlidbbal.GetNaturalLanguageRulesForSqlNodeTypeNnv(nnToken);
         if (ruleIdNnv != 0) // IF A VALID RULE EXIST
         {
          // GET THE TOKEN ID
          nnTokenId = nlidbbal.GetTokenIdInQuestion(questionId, nnToken);
          // SAVE TOKEN AND NATURAL LANGUAGE RULE TO DATABASE - THE NATURAL LANGUGAE
 RULE IS MAPPED TO THE SQL NODE TYPE. THEREFORE THE TOKEN IS MAPPED TO THE SQL NODE NNV
          nlidbbal.AddToTokensMappedToSqlNode(nnTokenId, ruleIdNnv);
         }
        }

        // GET THE COUNT OF TOKENS MAPPED TO NATURAL LANGUAGE RULES - THIS WILL ONLY
 EXTRACT NNT AND NNV TYPES
        int countOfTokensMappedToSqlNodes =
 nlidbbal.CountTokenToSqlNodeMappingForQuestion(questionId);
        // IF THERE IS ONLY ONE TOKEN MAPPED TO A TABLE OR VIEW
        if (countOfTokensMappedToSqlNodes == 1) {
         // CHECK FOR REQUESTED COLUMN NAMES BASED ON THE TOKEN NNT OR NNV
         // GET THE NNT OR NNV TOKEN
         string token =
 nlidbbal.GetNntOrNnvTokensMappedToSqlNodesDetailsForQuestion(questionId);

         List < string > dependentTokens = new List < string > ();
         // GET THE TOKENS DEPENDING ON THE NNT OR NNV TOKEN
         dependentTokens = nlidbbal.GetDependenciesOnTokens(questionId, token);

         // FOR ALL DEPENDENT TOKENS CHECK IF THERE ARE NATURAL LANGUAGE RULES DEFINED
 FOR SQL NODE TYPE NNC (NAME NODE COLUMN)
         foreach(var dependentToken in dependentTokens) {
          // GET THE NATURAL LANGUAGE RULE FOR NAME NODE VIEW
          ruleIdNnc = nlidbbal.GetNaturalLanguageRulesForSqlNodeTypeNnc(dependentToken);

          if (ruleIdNnc != 0) // IF A VALID RULE EXIST
          {
           // GET THE TOKEN ID
           nnTokenId = nlidbbal.GetTokenIdInQuestion(questionId, dependentToken);
           // SAVE TOKEN AND NATURAL LANGUAGE RULE TO DATABASE - THE NATURAL LANGUGAE
 RULE IS MAPPED TO THE SQL NODE TYPE. THEREFORE THE TOKEN IS MAPPED TO THE SQL NODE NNC
           nlidbbal.AddToTokensMappedToSqlNode(nnTokenId, ruleIdNnc);
          }
         }

         // GENERATE THE SQL STATEMENT FOR THE QUESTION IN NATURAL LANGUAGE
         string sqlStatement = nlidbbal.GenerateSqlStatement(questionId);

         nlidbbal.AddNewSqlQuestionToKB(questionId, sqlStatement);
         Console.WriteLine(sqlStatement);
        } else if (countOfTokensMappedToSqlNodes == 0) // THERE ARE NO TOKENS MAPPED TO
 TABLES OR VIEWS
        {
         Console.WriteLine(nlidbbal.UserMessageNoTablesOrViewImplementation);
         nlidbl.PrintMessage(nlidbbal.UserMessageNoTablesOrViewImplementation);
        } else // THERE ARE MULTIPLE TOKENS MAPPED TO TABLES OR VIEWS
        {
         Console.WriteLine(nlidbbal.UserMessageMultipleTablesOrViewImplementation);
         nlidbl.PrintMessage(nlidbbal.UserMessageMultipleTablesOrViewImplementation);
        }
       }
      } else // THE NUMBER OF SENTENCES HAS EXCEEDED THE PREDEFINED MAX_SENTENSES
      {
       string warningMessage =
 nlidbbal.GetWarningMessage_NumberOfSentenses(sentencesCount);
       Console.WriteLine(warningMessage);
       nlidbl.PrintMessage(warningMessage);
      }
     }
    } else // INPUT STRING IS EMPTY
    {
     Console.WriteLine(nlidbl.MessageEmptyString);
     nlidbl.PrintMessage(nlidbl.MessageEmptyString);
    }

    Console.ReadLine();
   }
  }
 }
```

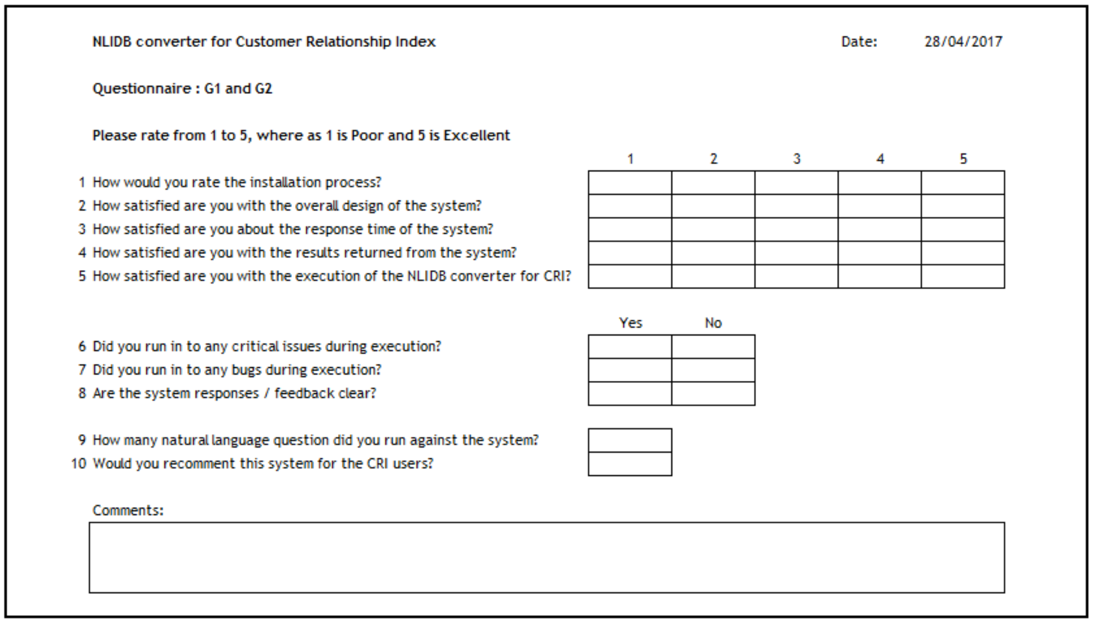
APPENDIX B – Stanford CoreNLP Annotators

Property name	Annotator class name	Description
tokenize	TokenizerAnnotator	Tokenizes the text.
ssplit	WordsToSentencesAnnotator	Splits a sequence of tokens into sentences.
pos	POSTaggerAnnotator	Labels tokens with their POS tag.
lemma	MorphaAnnotator	Generates the word lemmas for all tokens in the corpus.
ner	NERClassifierCombiner	Recognizes named (PERSON, LOCATION, ORGANIZATION, MISC), numerical (MONEY, NUMBER, ORDINAL, PERCENT), and temporal (DATE, TIME, DURATION, SET) entities.
parse	ParserAnnotator	Provides full syntactic analysis, using both the constituent and the dependency representations. The constituent-based output is saved in TreeAnnotation. The system generate three dependency-based outputs, as follows: basic, uncollapsed dependencies, saved in BasicDependenciesAnnotation; collapsed dependencies saved in CollapsedDependenciesAnnotation; and collapsed dependencies with processed coordinations, in

		CollapsedCCProcessedDependenciesAnnotation.
depparse	DependencyParseAnnotator	Provides a fast syntactic dependency parser. The system three dependency-based outputs, as follows: basic, uncollapsed dependencies, saved in BasicDependenciesAnnotation; collapsed dependencies saved in CollapsedDependenciesAnnotation; and collapsed dependencies with processed coordinations, in CollapsedCCProcessedDependenciesAnnotation. Most users of our parser will prefer the latter representation. For details about the dependency software, see this page. For more details about dependency parsing in general, see this page.
dcoref	DeterministicCorefAnnotator	Implements both pronominal and nominal coreference resolution. The entire coreference graph (with head words of mentions as nodes) is saved in CorefChainAnnotation.

APPENDIX C – Questionnaire

The questionnaire submitted to G1: Brandix Apparel Solution LTD – Essentials – Software Team and G2: Brandix Apparel Solution LTD – Essentials – Software Support Team



NLIDB converter for Customer Relationship Index

Date: 28/04/2017

Questionnaire : G1 and G2

Please rate from 1 to 5, where as 1 is Poor and 5 is Excellent

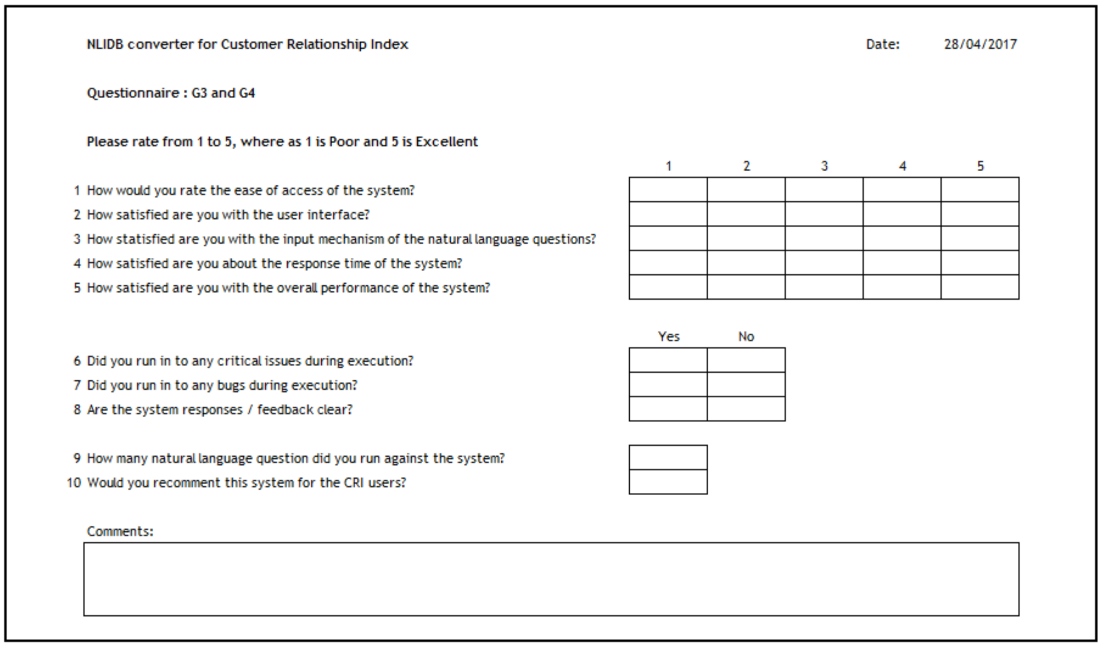
		1	2	3	4	5
1	How would you rate the installation process?					
2	How satisfied are you with the overall design of the system?					
3	How satisfied are you about the response time of the system?					
4	How satisfied are you with the results returned from the system?					
5	How satisfied are you with the execution of the NLIDB converter for CRI?					

		Yes	No
6	Did you run in to any critical issues during execution?		
7	Did you run in to any bugs during execution?		
8	Are the system responses / feedback clear?		

9	How many natural language question did you run against the system?	
10	Would you recomment this system for the CRI users?	

Comments:

The questionnaire submitted to G3: Customer Relationship Index User and G4: Non Customer Relationship Index Users.



NLIDB converter for Customer Relationship Index

Date: 28/04/2017

Questionnaire : G3 and G4

Please rate from 1 to 5, where as 1 is Poor and 5 is Excellent

		1	2	3	4	5
1	How would you rate the ease of access of the system?					
2	How satisfied are you with the user interface?					
3	How statisfied are you with the input mechanism of the natural language questions?					
4	How satisfied are you about the response time of the system?					
5	How satisfied are you with the overall performance of the system?					

		Yes	No
6	Did you run in to any critical issues during execution?		
7	Did you run in to any bugs during execution?		
8	Are the system responses / feedback clear?		

9	How many natural language question did you run against the system?	
10	Would you recomment this system for the CRI users?	

Comments: