Preposition Prediction in Writing Aids

Master's Project Report

May 2013

CMPT 697: Master's Project in Computer Science

Advisor:

Dr. Aparna Varde

Project Committee Members:

Dr. Anna Feldman

Dr. Emily Hill

Submitted by

Pooja Bhagat

Department of Computer Science
Montclair State University
Montclair, NJ 07043

Table of Contents

Abstract

Preposition is the word placed prior to the noun/adjective. Naturally, 'prepositions' in English play a key role in determining the meaning of a phrase or sentence. The computational analysis of prepositional usage has attracted significant attention. To predict accurately which preposition is most likely to occur in the given sentence is a challenging job. In particular, preposition usage is one of the most difficult aspects of English grammar for non-native speakers to master. The National Clearinghouse for English Language Acquisition (2002) estimates that 9.6% of the students in the US public school population speak a language other than English and have limited English proficiency. Clearly, there is a substantial and increasing need for tools for instruction in English as a Second Language (ESL). [2]

So, this project aims at implementing an approach for preposition prediction to develop a writing aid for ESL learners. Writing aid can help ESL learners to find most appropriate preposition missing in the sentence. It can serve as a useful and handy tool for ESL learners, which they can utilize to understand the use of the prepositions in the sentence.

We used the data driven approach to find the missing preposition in a sentence. Using different types of search techniques and pattern matching methods against large number of database records, we were able to find the missing preposition with some accuracy.

1. Introduction

1.1 Backgrounds and Motivation

To determine the correct preposition in the sentence is a challenging task for learners because prepositions do not have easily definable patterns which can be of assistance in making choices in novel contexts. In English, prepositions appear in adjuncts, they mark the arguments of predicates, and they combine with other parts of speech to express new meanings. The requirements of the language often seem entirely idiosyncratic and unpredictable even across nearly identical contexts. For example,

We say "*I work **in** Verona* but *I work **at** Unilever"*

despite the fact that the sentences have the same structure. Another potential pitfall is the fact that words with different POS but relating to the same lexical item will often require different prepositions, as in

"*Independent **of***" versus "*independent **from***".

Similarly, words that are related or have synonymous meanings cannot be relied on to follow the same prepositional patterns either, e.g.,

We say "*reason **for***" but "*cause **of**",* for example.

It is therefore not surprising that learners encounter problems in mastering preposition usage, and it is often equally hard for native speakers to articulate the reasons for these differences or offer guidance on how to overcome such problems.

The choice of preposition in an adjunct is largely constrained by its object

"**in** the summer", "**on** Friday", "**at** noon"

and the intended meaning

"**at** the beach", "**on** the beach", "**near** the beach", "**by** the beach".

Since adjuncts are optional and tend to be flexible in their position in a sentence, the task is quite complex for learners. [1]

In particular, preposition usage is one of the hardest aspects of English grammar for non-native speakers to master. The National Clearinghouse for English Language Acquisition (2002) estimates that 9.6% of the students in the US public school population speak a language other than English and have limited English fluency. Clearly, there is a substantial and increasing need for tools for instruction in English as a Second Language (ESL). [2]

1.2 Problem Definition

In order to describe the problem addressed in this paper, we first explain the term 'preposition'.[3]

A **preposition** may be defined as connecting word showing the relation of a noun or a noun substitute to some other word in the sentence (the squirrel in the tree; the preposition in shows the relationship between the squirrel and the tree.). Over ninety percent of preposition usage involves these nine prepositions:

with at by to in for
from of on

The main goal of this project is to find missing preposition from the sentence. When user will enter the sentence with missing preposition in the tool, it should return the correct preposition or prepositions (there may be more than one correct preposition in the sentence), which can fit in the sentence.

Example: sentence with missing preposition

John drove -------- Cambridge

Our tool should be able to return the correct answer: **to**

Prepositions and ESL

According to wiki **English as a second language** (**ESL**) is the use or study of English by speakers with different native languages. Language teaching practice often assumes that most of the difficulties that learners face in the study of English are a consequence of the degree to which their native language differs from English. English prepositions are a problem for learners because different languages use different prepositions to express the same ideas.

Preposition errors account for a significant proportion of all ESL grammar errors. They represented the largest category, about 29%, of all the errors by 53 intermediate to advanced ESL students (Bitchener et al., 2005), and 18% of all errors reported in an intensive analysis of one Japanese writer (Murata and Ishara, 2004). Preposition errors are not only prominent among error types; they are also quite frequent in ESL writing.

In particular, preposition usage is one of the most difficult aspects of English grammar for non-native speakers to master. The National Clearinghouse for English Language Acquisition (2002) estimates that 9.6% of the students in the US public school population speak a language other

than English and have limited English proficiency. Clearly, there is a substantial and increasing need for tools for instruction in English as a Second Language (ESL). [2]

This project aims at implementing an approach for preposition prediction to develop a writing aid for ESL learners. This writing aid can help them to find most appropriate preposition missing in the sentence. It can serve as a useful and handy tool for ESL learners, which they can utilize to understand the use of prepositions in the sentence.

1.3 Related Work

The previous work on the preposition prediction task varied in

1) The features selected
2) The number of prepositions tackled
3) The training and testing corpora used.

De Felice (2008) presents a system that (among other things) is used to predict the correct preposition for a given context. The system tackles the nine most frequent prepositions in English:

Of, to, in, for, on, with, at, by, from

The approach uses a wide variety of syntactic and semantic features: the lexical item modified by the PP, the lexical item that occurs as the object of the preposition, the POS tags of three words to the left and three words to the right of the preposition, the grammatical relation that the preposition is in with its object, the grammatical relation the preposition is in with the word modified by the PP, and the WordNet classes of the preposition's object and the lexical item modified by the PP. De Felice (2008) also used a named entity recognizer to extract generalizations about which classes of named entities can occur with which prepositions. Further, the verbs sub categorization frames were taken as features. For features that used lexical

sources (WordNet classes, verbs sub categorization frames), only partial coverage of the training and testing instances is available. [4]

Gamon et al. (2008) introduce a system for the detection of a variety of learner errors in nonnative English text, including preposition errors. For the preposition task, the authors combine the outputs of a classifier and a language model. The language model is a 5-gram model trained on the English Gigaword corpus. The classifier is trained 268 on Encarta encyclopedia and Reuter's news text. It operates in two stages: The presence/absence classifier predicts first whether a preposition needs to be inserted at a given location. Then, the choice classifier determines which preposition is to be inserted. The features that are extracted for each possible insertion site come from a six-token window around the possible insertion site. Those features are the relative positions, POS tags, and surface forms of the tokens in that window. The choice classifier predicts one of 13 prepositions:

in, for, of, on, to, with, at, by, as, from, since, about, than, and other [13]

Bergsma et al. (2009) extract contextual features from the Google 5-gram corpus to train an SVM-based classifier for predicting prepositions [12].

Tetreault and Chodorow (2008) present a system for detecting preposition errors in learner text. Their approach extracts a total of 25 features from the local contexts: the adjacent words, the heads of the nearby phrases, and the POS tags of all those. They combine word-based features with POS tag features to better handle cases where a word from the test instance has not been seen in training. For each test instance, the system predicts one of 34 prepositions [2].

Using a web-as-corpus approach, Elghfari et al. (2010) investigated the classification based solely on the relative number of occurrences for

target n- grams varying in preposition usage. They showed that such a surface-based approach is competitive with the published state-of-the-art results relying on complex feature sets. They stated that where enough data is available, in a surprising number of cases it thus is possible to obtain sufficient information for the relatively narrow window of context provided by n-grams which are small enough to frequently occur but large enough to contain enough predictive information about preposition usage.

They used section J of the BNC, as their test corpus. They selected all the sentences that contain one or more prepositions, using the POS annotation in the corpus to identify the prepositions. They marked every occurrence of these preposition tags in the corpus, yielding one prediction task for each marked preposition. They focused experiments on the top nine prepositions occurred in the BNC: of, to, in, for, on, with, at, by, from. For each occurrence of these nine prepositions in section J of the BNC, they extracted one prediction task, yielding a tag set of 522 313 instances. Due to various constrains and separating the training data and test data, they ended up with 8060 prediction tasks suitable for querying the Yahoo search engine. With 6166 test sets returning correct and 1894 returning incorrect, they could achieve a total accuracy of 76.5% [3].

2. Proposed Solution

Our approach for the defined problem is different than previous state-of-the-art approaches. A focus of the previous literature is on the question which linguistic and lexical features are the best predictors for preposition usage. Linguistic features used include the POS tags of the surrounding words, PP attachment sites, WordNet classes of PP object and modified item. Lexical features used include the object of the PP and the lexical item modified by the PP. Those syntactic, semantic and lexical features are then extracted

from the training instances and used by the machine-learning tool to predict the missing preposition in a test instance [3].

Instead of focusing on syntactic, semantic and lexical properties of surrounding words we directly use words themselves. We believe that these words around the prepositions are responsible to choose the preposition in that particular context. Surrounding words are also sufficient enough to define the context for prepositions. If enough data is available, we can predict the missing preposition in the sentence with a certain amount of accuracy. But of course, this is a large project and there are uncountable combinations available for words and prepositions so, the project has its own limitations. Extracting a large number of tokens from text corpus and then using these tokens to understand the use of prepositions is certainly a new and unique approach.

Our approach has the following steps

1. Extract tokens from text corpus (BNC)
2. Develop a tool where the user can enter the sentence with the missing preposition
3. Return a correct preposition or prepositions to user by making use of extracted tokens.

Step 1: Extract tokens from text corpus:

We used BNC text corpus as a training data.

BNC Corpus: The British National Corpus (BNC) is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of British English from the later part of the 20th century, both spoken and written. The latest edition is the *BNC XML Edition*, released in 2007. The written part of the BNC (90%) includes, for example, extracts from regional and national newspapers, specialist periodicals and journals for all ages and interests,

Academic books and popular fiction published and unpublished letters and memoranda, school and university essays, among many other kinds of text [9].

We automatically extracted preposition tokens from the text corpus and saved it in the database. Whenever we encountered the preposition in the sentence we extracted 2 words before the preposition, the preposition and two words after the preposition. After extracting these prepositions from the text corpus we saved it in the database. We used the British national corpus for token extraction. We worked on the A section of the BNC plain corpus.

Example:

Sentence:

While the people **of** Hong Kong, stripped **of** any useful British nationality **by** successive immigration laws, fear **for** their future **afte**r the massacre at Peking **in** June, many Macao citizens, just 40 miles away across the Pearl River delta, rest secure **in** the knowledge that Portugal will offer them a guaranteed home **of** last resort after it returns their territory **to** China **in** 1999.

We can see here in the sentence all the prepositions are in bold.

Following is the list of tokens, which we selected from this sentence.

1. the people **of** Hong Kong
2. stripped **of** any useful
3. British nationality **by** successive immigration laws
4. fear **for** their future
5. their future **afte**r the massacre
6. at Peking **in** June
7. guaranteed home **of** last resort
8. their territory **to** China

9. China **in** 1999

Here you can see the last token has 2 words before the preposition but has only one word after the preposition so we placed 0 at last position to match it with the other patterns.

We omitted all the punctuation marks and other symbols, which occurred in the text files while extracting tokens.

We selected the next word after the punctuation mark instead of the punctuation mark.

Example: laws, fear **for** their future

After omitting the punctuation mark: laws fear **for** their future

Step 2: Develop a tool where user can enter a query (sentence with missing preposition)

We developed a simple user interface, which ESL learners can use easily. We purposefully made it simple so any ESL learner with the limited knowledge of English also can use it easily.

Here, the user will enter the sentence with the missing preposition in the given text-box.

In the place of the missing preposition the user will enter -----

Example:

John ------ Cambridge

Here the user needs to give one space (press space key once) before ----- and one space (press space key once) after -------

After this, the tool automatically extracts two words before the missing preposition and two words after the missing preposition and checks the pattern against the database tokens to return the correct answer.

Step 3: Return a correct preposition or prepositions to user

Here the tool returns the exact preposition or prepositions that are suitable in the entered sentence. The tool uses various search methods over a preposition database that we develop by extracting relevant portions from the text corpora. When it finds the exact match for entered patterns it returns the answer to the user. We use pattern-matching techniques to search the exact match against database. As we can see certain prepositions have defined structure.

There are limited number of prepositions can be used after certain verbs or adjectives. After extracting large numbers of tokens from the text corpus we can find these repeated patterns easily.

Examples are shown below.

Uses of Prepositions after Certain Verbs

account for
agree on (something)
agree with (someone)
apologize to
apply for
approve of
argue with (someone)
ask for
believe in
belong to
blame (someone) for (something)
blame (something) on (someone)
borrow from
call on (upon)
care for
compliment (someone) on
come from
consent to
consist of
convince (someone) of (something)
decide on (upon)
listen for
listen to
look at
look for
look forward to
object to
plan on
provide for
provide with
recover from
remind (someone) of
search for
see about
substitute for
talk about
talk of
telephone to
think about
think of
wait for
wait on (meaning serve)

depend on (upon)
get rid of
hear about
hear from
hear of
insist on (upon)
invite (someone) to
laugh at

Uses of Prepositions with Certain Adjectives and in Idiomatic Expressions

according to
accustomed to
angry about (something)
capable of
composed of
content with
dependent on (upon)
different from (than)
disappointed in
due to
followed by
fond of
have respect for
in accordance with
angry at (someone)
angry with (someone)
based on
independent of
in regard to
interested in
limited to
married to
proud of
related to
resulting from
similar to
tired of

Where enough data is available, the tool can detect these repeated and definable patterns with the more accuracy [3] [7].

Other prepositions which do not have this kind of certain pattern and whose place is not sure after verbs and adjectives can be found by their repeated usage with different words and verbs.

If the same pattern of words repeating many times then we can say that this the correct pattern with the guarantee. Same approach is used here to find the correct prepositions in the sentence.

3. Implementation Details

We discuss the implementation of our approach along with a summary of our experimental evaluation.

3.1 System specifications:

We used Java Language for development of this tool and NetBeans 6.8 IDE for execution. The database software we used here is MySQL database provided by phpMyAdmin.

3.2 Preposition Database:

Developing a Preposition Database is an important part of this project. We used the BNC corpus as the training dataset. We automatically extracted 5-grams from the text file and saved it in the database. We created a large database to provide the tool lot of combination of words, which can help it to find the exact match for entered word patterns.

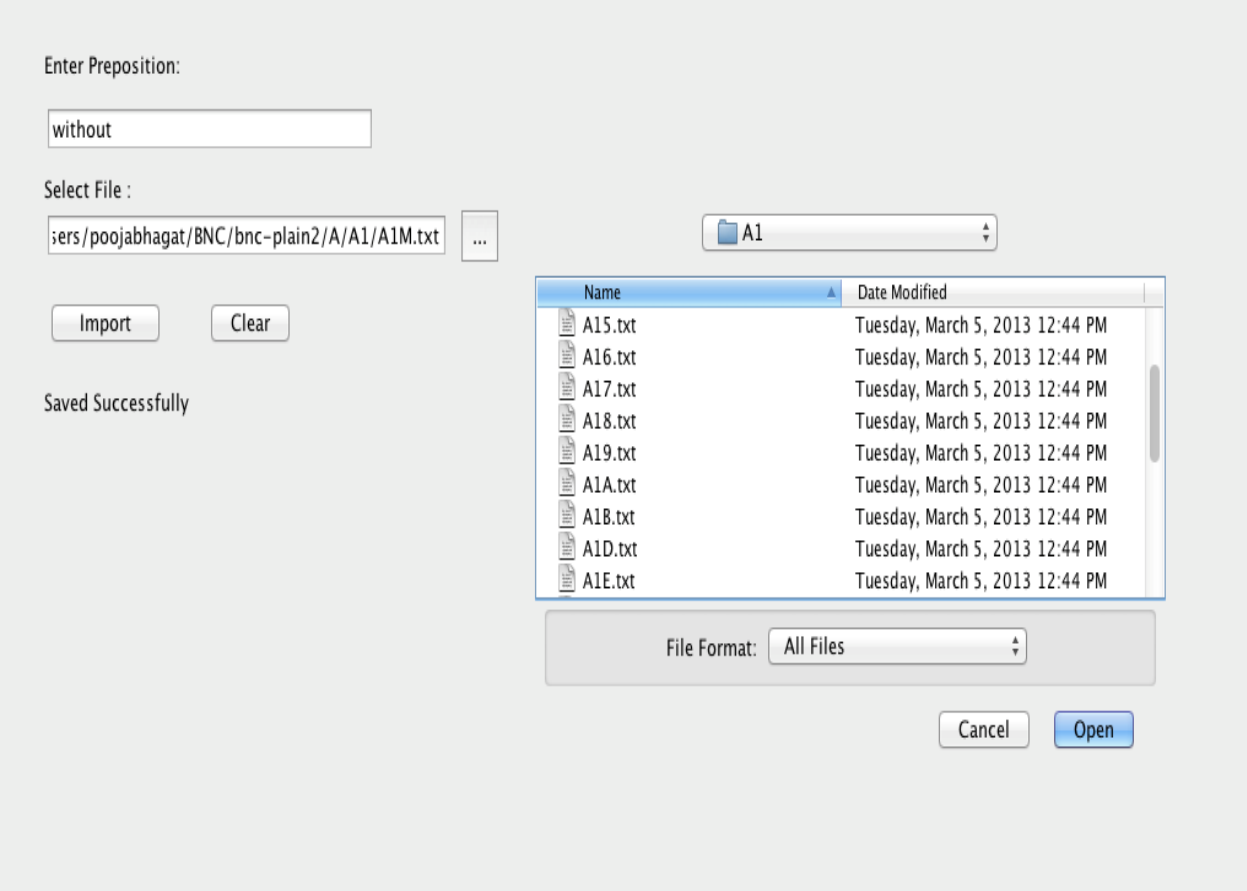

Figure1. Importing tokens from the text corpus

We can see here in Figure 1 the screen whenever we enter the desired preposition in the first text box and select the text file we want to work on and press the 'Import' button, system automatically extracts tokens containing entered preposition and saves it in the database.
The actual database has one table and 5 fields, where the third field is the preposition and remaining fields have words surrounding the preposition.

word1	word2	word3	word4	word5
had	changed	since	May	1979
0	133	since	it	seems
have	seen	since	1969	0
first	time	since	he	became
at	risk	through	our	continued
to	get	through	military	cordons
be	punished	throughout	life	for
and	wait	till	the	first
is	expected	to	concede	defeat
fundamentalists	threaten	to	fight	fight
a	willingness	to	surrender	much
was	likely	to	become	mandatory
0	16	to	list	the
policies	is	to	invite	boredom
are	coming	to	accept	that
battle	likely	to	prove	appealing
prove	appealing	to	the	sort
for	Labour	to	win	on
selling	point	to	challenge	the
went	on	to	say	that
recent	convert	to	revisionism	0
a	mandate	to	build	a
dont	seem	to	have	the
answer	is	to	combine	a

Figure 2. Preposition Database

Here we can see in Figure 2 the prepositions and the words surrounding the prepositions. Wherever system doesn't find any word it puts 0 in place of missing words. The system also avoids punctuation marks and symbols, which are present in the text file.

Here are some symbols, which we avoided storing in the database.

, : ? / " " ' ' ; { } [ ] ( ) # ^ * | \ +

We select immediate next word after these characters. Currently the database has thousands of records.
More the number of the database records more accurate are the results.

3.3 The Preposition Tool

3.3.1 Basic Model

We start with the simple interface and simple technique to find the missing preposition. Here the user needs to manually enter the sentence and 2 words before and after the preposition. In the text box with the label 'Enter preposition' user will enter any random preposition, if it is the correct fit for the entered sentence then count will change. Otherwise the count will remain 0. We used some other search conditions also. For example: Search Left one Right one: Here we search one word before the preposition, the preposition and one word after the preposition. If we find an exact match for this combination in database the count will change and we will get correct preposition. This basic model helped for the initial experiments we performed on the data. It is illustrated in Figure 3.

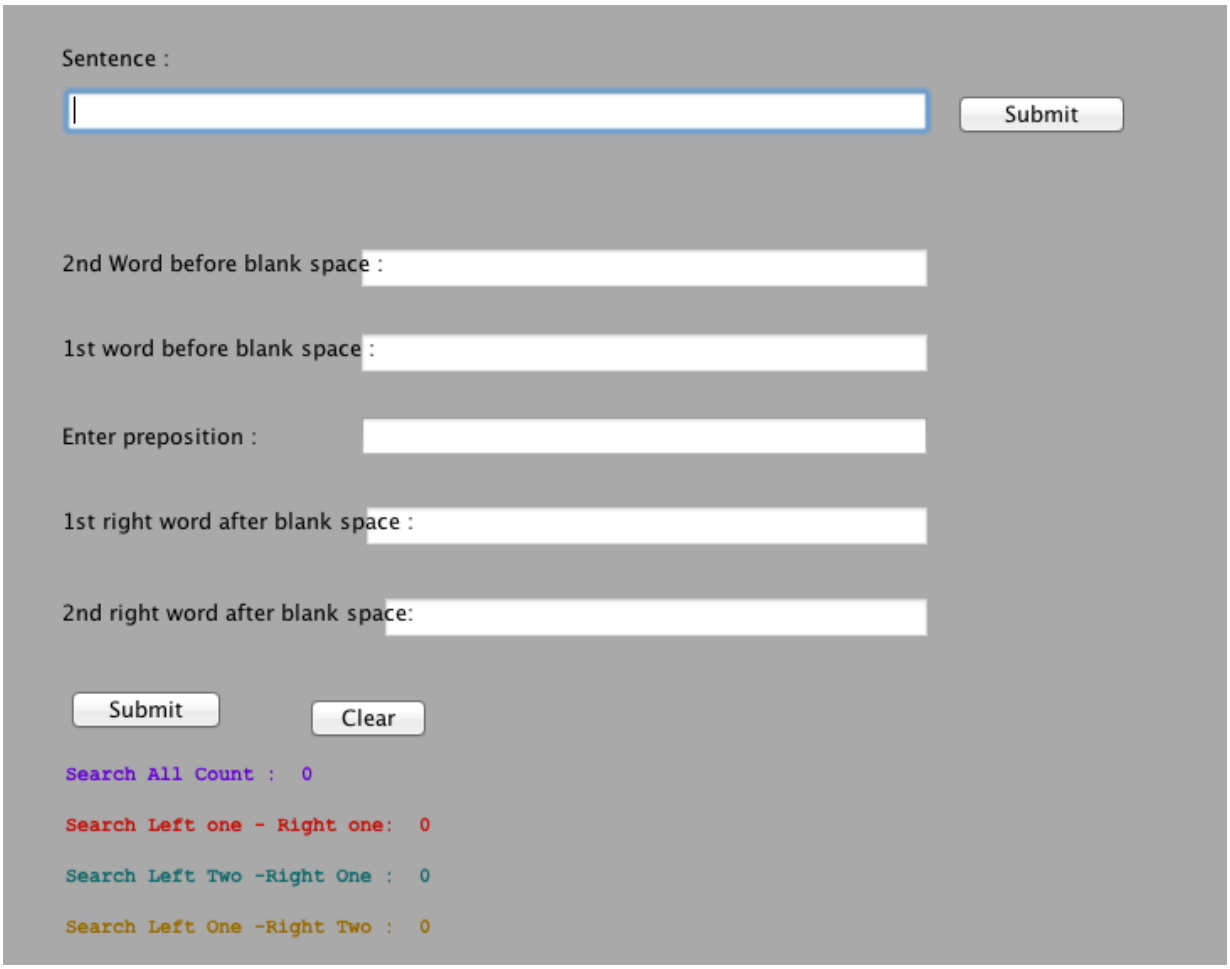

Figure 3. Basic Preposition Tool

3.3.2 Advanced Model

In advanced model we automate most of the preposition prediction task. When the user enters the sentence in the tool system automatically extracts two words before and two words after the preposition and searches the extracted word patterns against the database. If the pattern matches with available tokens in the database then the tools returns one or more prepositions. We use the following search conditions for preposition prediction.

1. Search all Counts:

Search the pattern in the database for all extracted four words, i.e., two words before the blank space and two words after the preposition. If that four-word pattern matches with any record available in the database, then the tool returns the prepositions, which are present in those records.

2. Search Left one - Right one:

Here we search only one word before and one word after the blank space. If this two-words pattern matches with the any available records in the database then system returns the prepositions from matched records.

3. Search Left Two - Right one:

Here we search two words before the blank space and only word after the blank space and search this pattern against the database to find the prepositions.

4. Search Left One - Right Two:

Here we select one word before blank space and two words after the blank space and search the pattern against the database to find the prepositions.

5. Search Left Two:

Here we select two words before the blank space and search the pattern against the database to find the prepositions.

6. Search Right Two

Here we select two words after the blank space and search the pattern against the database to find the prepositions.

We can see the working of this tool with example:

Enter the sentence: I start work ------- eight in the morning

The Image 4 shows how the tool returns the correct preposition and its total number of occurrences in three search conditions – Search Left one-right one, Search Left Two and Search Left one-Right Two

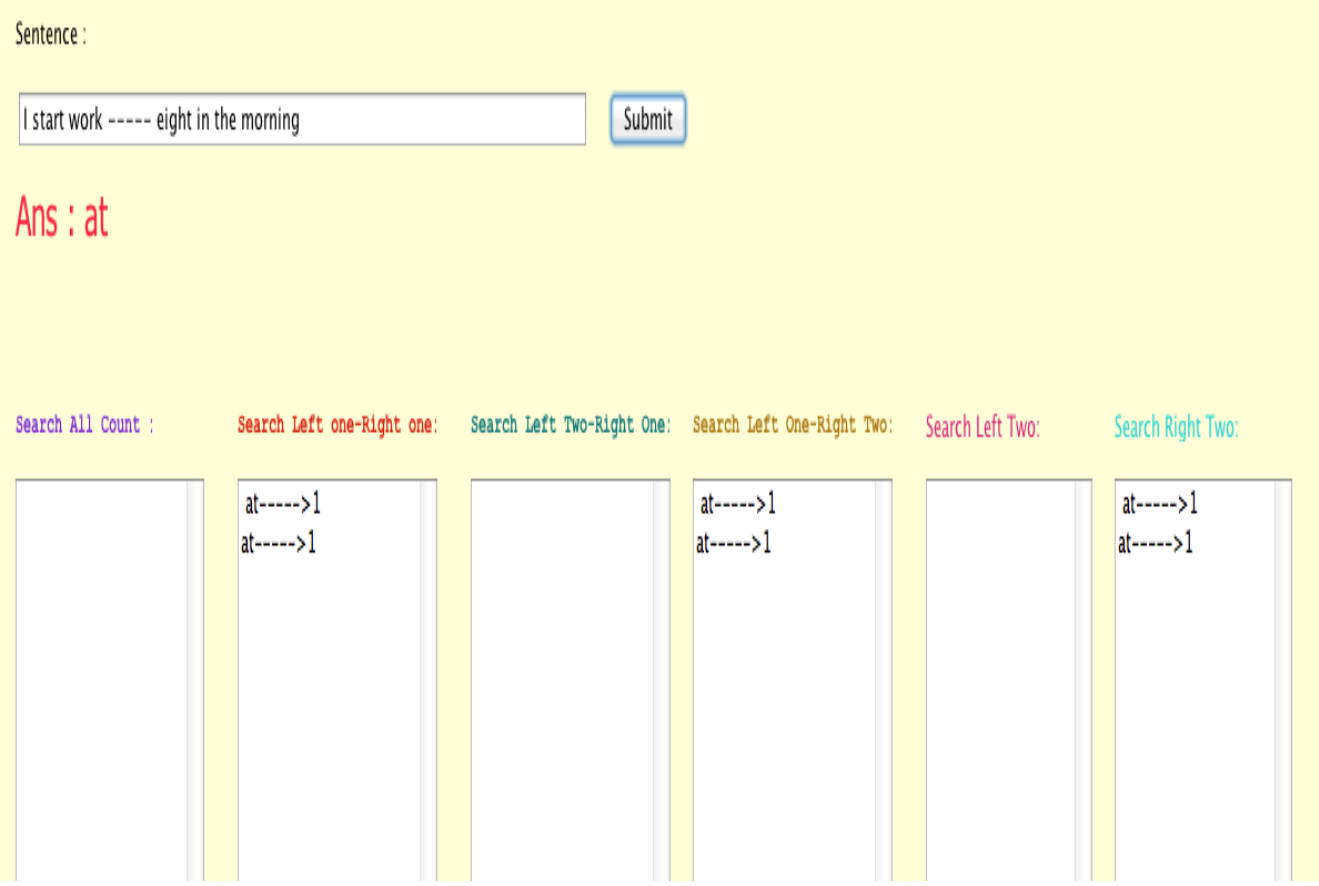

Image 4:Advanced Preposition Tool

4. Test Results

We extracted the database tokens from the A section of the BNC. So we tested the tool against D section of the BNC. The D section of the BNC has different sentences that A section. We marked all the occurrences of the preposition 'to' in text file and entered all these sentences with the preposition 'to' in the tool one by one. We came up with some interesting results. The tool returned few correct answers but sometimes tool returned more than one answers. In those multiple answers sometimes there was a correct answer. In a few cases the system did not encounter any answer at all. From this experience we can say that this is the pilot model for preposition prediction. We can keep improvising the tool in coming future.

5. Conclusions and Future Work

We have addressed the issue of finding missing preposition in the sentence. We used pattern matching technique and data driven approach to overcome this issue.

In this work we make the following contributions:

1. Proposing a methodology to discover the missing preposition from any given sentence.
2. Implementing a solution with a pattern matching technique over the abstracted n-grams with experimentation over real corpus and develop an ESL tool for ESL learners
3. Developing a database of preposition for future use in preposition related experiments.

Future work in this area includes the following.

1.Using more training data: Even a cursory examination of the training corpus reveals that there are many gaps in the data. 75k seem like a large number of examples, but the selection of prepositions is highly dependent on the presence of other specific words in the context. Many fairly common combinations of Verb+Preposition+Noun or Noun+Preposition+Noun are simply not attested, even in a sizable corpus. That is, prediction is better for prepositions that have many examples in the training set and worse for those with fewer examples. This suggests the need for much more data.

2.Distinguishing between lexical and functional prepositions: It could be useful to distinguish between lexical and functional prepositions when returning results. This is an important distinction because the information needed to predict functional prepositions typically is in the local context, whereas the information needed to predict lexical prepositions is not necessarily present locally. To illustrate, a competent human speaker presented with the sentence **John is dependent -----his brother** and asked to fill in the missing preposition, would correctly pick **on**. This is a case of a functional preposition where the relevant information is locally present: the adjective dependent selects on. On the other hand, the sentence **John put his bag ------ the table** is more problematic, even for a human, since both **on** and **unde**r are reasonable choices; the information needed to predict the omitted preposition in this case is not locally present. In this project we treated these both different types of sentence classes alike. But in future, we can differentiate between these two classes and test the tool separately on two different types of sentences and check the results are better or not?

3.Currently, tool is working on 5-grams but it would be interesting to check the tool against 7-gram tokens extracted from the text corpus.

4.Instead of working on only word-based features, someone would like to work on POS tags of words surrounding the preposition. It will help to reduce the size of the database.
5.Also, it would be interesting to see how tool work on any other corpus than BNC.

Acknowledgments

I would like to express my special gratitude and thanks to all the people who helped me complete this project.

I would like to thank Dr. Aparna Varde, my Project Advisor for her constant support, help and guidance. She motivated me at every step of my project. All the books, web sites and other material that she made available for me were very useful for my project.

I would like to thank my project committee members specially Dr. Anna Feldman for her invaluable inputs and comments on my project. She helped me to keep my focus on the end goal.

In general, I would like to thank the department chair Dr.Katherine Herbert, all graduate committee members, faculty, staff and students in Department of Computer Science at Montclair State University and my family for their co-operation.

References

[1] De Felice, Rachele. 2008. Automatic Error Detection in Non-native English. Ph.D. Thesis, St Catherine's College, University of Oxford.

[2] Chodorow, Martin, Joel Tetreault, and Na-Rae Han.2007. Detection of grammatical errors involving prepositions. In Proceedings of the 4th ACLSIGSEM Workshop on Prepositions, pages 25–30,Prague, Czech Republic, June.

[3] Anas Elghafari, Detmar Meurers and Holger Wunsch. 2010. Exploring the Data-Driven Prediction of Prepositions in English. In COLING'10 .The 23rd International Conference on Computational Linguistics, Pages 267-275, Stroudsburg, PA, USA, 2010

[4] Rachele de Felice and Stephen J. Pulman.2007.Automatically acquiring models of preposition use. In SigSem'07 The Fourth ACL-SIGSEM workshop on Prepositions, Pages 45-50,Strousburg, PA, USA.

[5] ESL info Available from:
http://en.wikipedia.org/wiki/English_as_a_second_or_foreign_language

[3] Prepositions Info available from:
http://en.wikipedia.org/wiki/Preposition_and_postposition

[6] Daniel Jurafsky and James H. Martin.2008. Speech and Language Processing
http://www.cs.colorado.edu/~martin/SLP/Updates/1.pdf

[7]Preposition info available from
http://rwc.hunter.cuny.edu/reading-writing/on-line/prep-def.html

[8]Database:
http://localhost/phpmyadmin/index.php?db=dbset&token=61ceb46c271d9a2d4378438813032539

[9] BNC info available from:

http://www.natcorp.ox.ac.uk/corpus/index.xml

[11] NetBeans IDE available from:

https://netbeans.org/community/releases/68/install.html

[12] Bergsma, Shane, Dekang Lin, and Randy Goebel.2009. Web-scale n-gram models for lexical disambiguation. In IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence,pages 1507–1512, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[13] Gamon, Michael, Jianfeng Gao, Chris Brockett,Alexander Klementiev, William Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using

contextual speller techniques and language modeling for esl error correction. In Proceedings of IJCNLP, Hyderabad, India.

Appendix: Programs (Java Code)

The following is the list of all the programs that are part of the application developed during the course of this project.

ImportToDB.java

```
*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * ImportToDB.java
 *
 * Created on Apr 6, 2013, 2:34:40 PM
 */

package preposiontool;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.sql.*;
import javax.swing.JOptionPane;
/**
 *
 * @author gauri
 */
public class ImportToDB extends javax.swing.JFrame {
    Connection cn;
    Statement stmt;
    ResultSet rs;
    String sql;
    int cnt=0;
    String url ="jdbc:mysql://localhost/dbset";
    /** Creates new form ImportToDB */
    public ImportToDB() {
        initComponents();
        jFileChooser1.setVisible(false);
        try
        {
                Class.forName("com.mysql.jdbc.Driver").newInstance();
```

```
            cn=DriverManager.getConnection(url,"pooja","poojabhagat");


      }
      catch(Exception ex)
      {
         JOptionPane.showMessageDialog(this, ex.getMessage());
      }
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

      jButton1 = new javax.swing.JButton();
      LblMsg = new javax.swing.JLabel();
      jFileChooser1 = new javax.swing.JFileChooser();
      jLabel1 = new javax.swing.JLabel();
      jTextField1 = new javax.swing.JTextField();
      jButton2 = new javax.swing.JButton();
      jLabel2 = new javax.swing.JLabel();
      jTextField2 = new javax.swing.JTextField();
      jButton3 = new javax.swing.JButton();

      setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
      getContentPane().setLayout(null);

      jButton1.setText("Import");
      jButton1.addActionListener(new java.awt.event.ActionListener() {
         public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
         }
      });
      getContentPane().add(jButton1);
      jButton1.setBounds(50, 180, 100, 23);

      LblMsg.setText("-------");
      getContentPane().add(LblMsg);
      LblMsg.setBounds(50, 230, 880, 14);

      jFileChooser1.addActionListener(new java.awt.event.ActionListener() {
         public void actionPerformed(java.awt.event.ActionEvent evt) {
            jFileChooser1ActionPerformed(evt);
         }
      });
      getContentPane().add(jFileChooser1);
      jFileChooser1.setBounds(450, 120, 514, 330);
```

```
    jLabel1.setText("Select File :");
    getContentPane().add(jLabel1);
    jLabel1.setBounds(50, 110, 70, 14);
    getContentPane().add(jTextField1);
    jTextField1.setBounds(50, 130, 330, 20);

    jButton2.setText("...");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });
    getContentPane().add(jButton2);
    jButton2.setBounds(390, 130, 30, 23);

    jLabel2.setText("Enter Word :");
    getContentPane().add(jLabel2);
    jLabel2.setBounds(50, 40, 90, 14);
    getContentPane().add(jTextField2);
    jTextField2.setBounds(50, 70, 270, 20);

    jButton3.setText("Clear");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });
    getContentPane().add(jButton3);
    jButton3.setBounds(180, 180, 57, 23);

    pack();
}// </editor-fold>

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jFileChooser1.setVisible(true);
}

private void jFileChooser1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField1.setText(jFileChooser1.getSelectedFile().getAbsolutePath());
    jFileChooser1.setVisible(false);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try
    {
        String word=jTextField2.getText().toLowerCase().trim();
    File f=new File(jTextField1.getText());
    BufferedReader br = new BufferedReader(new FileReader(f));
    String line;
    while ((line = br.readLine()) != null)
```

```
{
   char [] ch={'.',',', ':','?','/',"","'",'\',';','{','}','[',']','(',')','#','^','*','|','\\','+'};
   for(int x=0;x<ch.length;x++)
   {
      line=line.replace(ch[x], ' ');
   }
   String []data=line.split(" ");
   for(int i=0;i<data.length;i++)
   {
      if(data[i].equalsIgnoreCase(word))
      {
         String w1="0",w2="0",w3="0",w4="0",w5="0";
         try
         {
            w1=data[i-2];
         }
         catch(Exception ex)
         {
            w1="0";
         }

         try
         {
            w2=data[i-1];
         }
         catch(Exception ex)
         {
            w2="0";
         }

         try
         {
            w3=word;
         }
         catch(Exception ex)
         {
            w3="0";
         }

         try
         {
            w4=data[i+1];
         }
         catch(Exception ex)
         {
            w4="0";
         }
         try
         {
            w5=data[i+2];
         }
         catch(Exception ex)
         {
```

```
                    w5="0";
                }
                try
                {
                String sql="insert into words1
values('"+w1+"','"+w2+"','"+w3+"','"+w4+"','"+w5+"')";
                stmt=cn.createStatement();
                stmt.executeUpdate(sql);
                }
                catch(Exception ex)
                {}
            }
        }

      }
      br.close();
      LblMsg.setText("Saved Successfully");
      }
      catch(Exception ex)
      {
      LblMsg.setText(ex.getMessage());
      }
   }

   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      jTextField1.setText("");
      jTextField2.setText("");
      LblMsg.setText("");
   }

   /**
   * @param args the command line arguments
   */
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new ImportToDB().setVisible(true);
         }
      });
   }

   // Variables declaration - do not modify
   private javax.swing.JLabel LblMsg;
   private javax.swing.JButton jButton1;
   private javax.swing.JButton jButton2;
   private javax.swing.JButton jButton3;
   private javax.swing.JFileChooser jFileChooser1;
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JTextField jTextField1;
   private javax.swing.JTextField jTextField2;
   // End of variables declaration
```

```
}
```

Main.java

PreProcess.java

ToolScreen.java

ToolWindow.java

```
package preposiontool;
import java.sql.*;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.Hashtable;
import javax.swing.JOptionPane;
/**
 *

 */
public class ToolWindow extends javax.swing.JFrame {

    String left2,left1,right1,right2;
    Connection cn;
          Statement stmt;
          ResultSet rs;
          String sql;
          int cnt=0;
        String url ="jdbc:mysql://localhost/dbset";

        Hashtable<String, String> allwords=new Hashtable<String, String>();
         Hashtable<String, String> leftonerightone=new Hashtable<String, String>();
          Hashtable<String, String> lefttworightone=new Hashtable<String, String>();
           Hashtable<String, String> leftonerighttwo=new Hashtable<String, String>();

            Hashtable<String, String> lefttwo=new Hashtable<String, String>();
           Hashtable<String, String> righttwo=new Hashtable<String, String>();

    /** Creates new form ToolWindow */
    public ToolWindow() {
        initComponents();
        try
                    {
                            Class.forName("com.mysql.jdbc.Driver").newInstance();
                       cn=DriverManager.getConnection(url,"pooja","poojabhagat");


                    }
                    catch(Exception ex)
                    {
                    JOptionPane.showMessageDialog(this, ex.getMessage());
```

```
            }
    }
boolean CheckSpace(String word)
    {
        word=word.toLowerCase();
        for(int j=0;j<word.length();j++)
        {
            char ch=word.charAt(j);
            if((ch!='-')&&(ch!='_'))
            {
                return false;
            }
        }
        return true;
    }
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        TxtSentence = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jLabel9 = new javax.swing.JLabel();
        list1 = new java.awt.List();
        list2 = new java.awt.List();
        list3 = new java.awt.List();
        list4 = new java.awt.List();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        list5 = new java.awt.List();
        jLabel5 = new javax.swing.JLabel();
        list6 = new java.awt.List();
        LblResult = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

        jLabel1.setText("Sentence :");
        getContentPane().add(jLabel1);
        jLabel1.setBounds(70, 20, 257, 14);
        getContentPane().add(TxtSentence);
        TxtSentence.setBounds(70, 40, 517, 29);

        jButton1.setText("Submit");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jButton1ActionPerformed(evt);
    }
});
getContentPane().add(jButton1);
jButton1.setBounds(680, 40, 100, 23);

jLabel9.setFont(new java.awt.Font("Courier New", 1, 12));
jLabel9.setForeground(new java.awt.Color(102, 0, 204));
jLabel9.setText("Search All Count :");
getContentPane().add(jLabel9);
jLabel9.setBounds(70, 120, 126, 14);
getContentPane().add(list1);
list1.setBounds(70, 150, 170, 310);
getContentPane().add(list2);
list2.setBounds(270, 150, 180, 310);
getContentPane().add(list3);
list3.setBounds(480, 150, 180, 310);
getContentPane().add(list4);
list4.setBounds(680, 150, 180, 310);

jLabel2.setFont(new java.awt.Font("Courier New", 1, 12));
jLabel2.setForeground(new java.awt.Color(204, 0, 0));
jLabel2.setText("Search Left one-Right one:");
getContentPane().add(jLabel2);
jLabel2.setBounds(270, 120, 182, 14);

jLabel3.setFont(new java.awt.Font("Courier New", 1, 12));
jLabel3.setForeground(new java.awt.Color(0, 102, 102));
jLabel3.setText("Search Left Two-Right One:");
getContentPane().add(jLabel3);
jLabel3.setBounds(480, 120, 190, 14);

jLabel10.setFont(new java.awt.Font("Courier New", 1, 12));
jLabel10.setForeground(new java.awt.Color(153, 102, 0));
jLabel10.setText("Search Left One-Right Two:");
getContentPane().add(jLabel10);
jLabel10.setBounds(680, 120, 190, 14);

jLabel4.setForeground(new java.awt.Color(204, 0, 102));
jLabel4.setText("Search Left Two:");
getContentPane().add(jLabel4);
jLabel4.setBounds(890, 120, 130, 14);
getContentPane().add(list5);
list5.setBounds(890, 150, 150, 310);

jLabel5.setForeground(new java.awt.Color(0, 204, 204));
jLabel5.setText("Search Right Two:");
getContentPane().add(jLabel5);
jLabel5.setBounds(1060, 120, 140, 14);
getContentPane().add(list6);
list6.setBounds(1060, 150, 160, 310);

LblResult.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
```

```
    LblResult.setForeground(new java.awt.Color(255, 0, 51));
    LblResult.setText("Ans :");
    getContentPane().add(LblResult);
    LblResult.setBounds(70, 80, 520, 17);

    pack();
}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    list1.removeAll();
    list2.removeAll();
    list3.removeAll();
    list4.removeAll();
    list5.removeAll();
    list6.removeAll();

    String sentence=TxtSentence.getText();
    String words[]=sentence.split(" ");
    for(int i=0;i<words.length;i++) {

        words[i]=words[i].trim();
    }
    for(int i=0;i<words.length;i++) {
        String word=words[i];
        //JOptionPane.showMessageDialog(this, word);
        boolean isspace=CheckSpace(word);
        if(isspace) {
            try {
                left1=(words[i-1]);
            } catch(Exception ex){}
            try {
                left2=(words[i-2]);
            } catch(Exception ex){}
            try {
                right1=(words[i+1]);
            } catch(Exception ex){}
            try {
                right2=(words[i+2]);
            } catch(Exception ex){}
            //return;
        }
    }
    ////////////////////////////////////////////////////////////////////////////////
     String word1,word2,word3,word4,word5;
     allwords.clear();
     leftonerightone.clear();
     lefttworightone.clear();
     leftonerighttwo.clear();
     lefttwo.clear();
     righttwo.clear();

     try {
```

```
        sql="select distinct(word3) from words1";
        stmt=cn.createStatement();
        rs=stmt.executeQuery(sql);
         while (rs.next()) {
               word3=rs.getString(1);
               allwords.put(word3, "0");
               leftonerightone.put(word3, "0");
               lefttworightone.put(word3, "0");
               leftonerighttwo.put(word3, "0");
              lefttwo.put(word3, "0");
              righttwo.put(word3, "0");

           }


        sql="select * from words1";

        try
        {

           stmt=cn.createStatement();
           rs=stmt.executeQuery(sql);
           while (rs.next()) {
              word1=rs.getString(1);
              word2=rs.getString(2);
              word3=rs.getString(3);
              word4=rs.getString(4);
              word5=rs.getString(5);

              try{

              if((left2.equalsIgnoreCase(word1.trim())) &&
(left1.equalsIgnoreCase(word2.trim()))&&
(right1.equalsIgnoreCase(word4.trim()))&&(right2.equalsIgnoreCase(word5.trim())))
              {
                 String cnt=allwords.get(word3);
                 int count=Integer.parseInt(cnt);
                 ++count;
                 allwords.put(word3, ""+count);
              }
              }
              catch(Exception ex){}

              try{
              if((left1.equalsIgnoreCase(word2.trim()))&&
(right1.equalsIgnoreCase(word4.trim())))
              {
                 String cnt=allwords.get(word3);
                 int count=Integer.parseInt(cnt);
                 ++count;
                 leftonerightone.put(word3, ""+count);
              }
              }
```

```
            catch(Exception ex){}

            try{
            if((left2.equalsIgnoreCase(word1.trim())) &&
(left1.equalsIgnoreCase(word2.trim()))&& (right1.equalsIgnoreCase(word4.trim())))
            {
               String cnt=allwords.get(word3);
               int count=Integer.parseInt(cnt);
               ++count;
               lefttworightone.put(word3, ""+count);
            }
         }
           catch(Exception ex){}
            try{
            if((left1.equalsIgnoreCase(word2.trim()))&&
(right1.equalsIgnoreCase(word4.trim()))&&(right2.equalsIgnoreCase(word5.trim())))
            {
               String cnt=allwords.get(word3);
               int count=Integer.parseInt(cnt);
               ++count;
               leftonerighttwo.put(word3, ""+count);
            }
            }
            catch(Exception ex){}
            try{
            if((left2.equalsIgnoreCase(word1.trim())) &&
(left1.equalsIgnoreCase(word2.trim())))
            {
               String cnt=lefttwo.get(word3);
               int count=Integer.parseInt(cnt);
               ++count;
               lefttwo.put(word3, ""+count);
            }
            }
            catch(Exception ex){}
            try{

if((right1.equalsIgnoreCase(word4.trim()))&&(right2.equalsIgnoreCase(word5.trim())))
            {
               String cnt=righttwo.get(word3);
               int count=Integer.parseInt(cnt);
               ++count;
               righttwo.put(word3, ""+count);
            }
            }
            catch(Exception ex){}
         }
      }catch(Exception ex){}

         ArrayList foundwords=new ArrayList();
         Enumeration<String>keys= allwords.keys();
         while(keys.hasMoreElements())
         {
```

```
   String key=keys.nextElement();
   String val=allwords.get(key);
   if(Integer.parseInt(val)>0)
   {
      list1.add(key+"----->"+val);
      if(foundwords.contains(key)==false)
         foundwords.add(key);
   }
}

Enumeration<String>keys1= leftonerightone.keys();
while(keys1.hasMoreElements())
{
   String key=keys1.nextElement();
   String val=leftonerightone.get(key);
   if(Integer.parseInt(val)>0)
   {
      list2.add(key+"----->"+val);
      if(foundwords.contains(key)==false)
         foundwords.add(key);
   }

}


Enumeration<String>keys2= lefttworightone.keys();
while(keys2.hasMoreElements())
{
   String key=keys2.nextElement();
   String val=lefttworightone.get(key);
   if(Integer.parseInt(val)>0)
   {
      list3.add(key+"----->"+val);
      if(foundwords.contains(key)==false)
         foundwords.add(key);
   }
}

Enumeration<String>keys3= leftonerighttwo.keys();
while(keys3.hasMoreElements())
{
   String key=keys3.nextElement();
   String val=leftonerighttwo.get(key);
   if(Integer.parseInt(val)>0)
   {
      list4.add(key+"----->"+val);
      if(foundwords.contains(key)==false)
         foundwords.add(key);
   }
}

Enumeration<String> keys4= lefttwo.keys();
while(keys4.hasMoreElements())
```

```
{
   String key=keys4.nextElement();
   String val=lefttwo.get(key);
   if(Integer.parseInt(val)>0)
   {
      list5.add(key+"----->"+val);
      if(foundwords.contains(key)==false)
         foundwords.add(key);
   }
}

Enumeration<String>keys5= righttwo.keys();
while(keys5.hasMoreElements())
{
   String key=keys5.nextElement();
   String val=righttwo.get(key);
   if(Integer.parseInt(val)>0)
   {
      list6.add(key+"----->"+val);
      if(foundwords.contains(key)==false)
         foundwords.add(key);
   }
}

try {

   String [][] data=new String[foundwords.size()][6];
   for(int i=0;i<foundwords.size();i++)
   {
      for(int j=0;j<6;j++)
      {
         data[i][j]="0";
      }
   }
   for(int i=0;i<foundwords.size();i++)
   {
      String searchword=foundwords.get(i).toString();
      String[] l1=list1.getItems();

      for(String s : l1)
      {
         if(s.contains(searchword))
         {
            String cnt=s.replace(searchword+"----->", "");
            data[i][0]=cnt;
            break;
         }
      }

      String[] l2=list2.getItems();
      for(String s : l2)
      {
         if(s.contains(searchword))
```

```
        {
            String cnt=s.replace(searchword+"----->", "");
            data[0][1]=cnt;
            break;
        }
    }

    String[] l3=list3.getItems();
    for(String s : l3)
    {
        if(s.contains(searchword))
        {
            String cnt=s.replace(searchword+"----->", "");
            data[0][2]=cnt;
            break;
        }
    }

    String[] l4=list4.getItems();
    for(String s : l4)
    {
        if(s.contains(searchword))
        {
            String cnt=s.replace(searchword+"----->", "");
            data[0][3]=cnt;
            break;
        }
    }

     String[] l5=list5.getItems();
    for(String s : l5)
    {
        if(s.contains(searchword))
        {
            String cnt=s.replace(searchword+"----->", "");
            data[0][4]=cnt;
            break;
        }
    }

     String[] l6=list6.getItems();
    for(String s : l6)
    {
        if(s.contains(searchword))
        {
            String cnt=s.replace(searchword+"----->", "");
            data[0][5]=cnt;
            break;
        }
    }

}
```

```
            String correctword="";
            int fnoofloc=0;
            int ftotamt=0;
            String [][]result=new String[foundwords.size()][4];
            for(int i=0;i<foundwords.size();i++)
            {
               String word=foundwords.get(i).toString();
               int noofloc=0;
               int totamt=0;
               for(int j=0;j<6;j++)
               {
                  String d=data[i][j];
                  if(Integer.parseInt(d)>0)
                     ++noofloc;
                  totamt+=Integer.parseInt(d);

               }
               result[i][0]=word;
               result[i][1]=""+noofloc;
               result[i][2]=""+totamt;

            }
            int pos=0;
            int maxcnt=0;
            for(int i=0;i<foundwords.size();i++)
            {
               int cnt=Integer.parseInt(result[i][1]);
               if(cnt>maxcnt)
               {
                  maxcnt=cnt;
                  pos=i;
               }
            }
            LblResult.setText("Ans : "+result[pos][0]);


         }
         catch(Exception ex){}



      } catch(Exception ex){
         JOptionPane.showMessageDialog(this, ex.getMessage());
      }

   }

   /**
   * @param args the command line arguments
   */
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
```

```
            new ToolWindow().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JLabel LblResult;
private javax.swing.JTextField TxtSentence;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel9;
private java.awt.List list1;
private java.awt.List list2;
private java.awt.List list3;
private java.awt.List list4;
private java.awt.List list5;
private java.awt.List list6;
// End of variables declaration

}
```

Words.java