

Git Commit Message Analysis – Appendices

Contents

Appendix A – code listing	1
Part 1 – main code	2
CommitMessage.java	2
MessageTag.java	5
MessageTagger.java	5
MessageGetter.java	6
BracketsFixer.java	8
TokenChecker.java	9
VagueNounChecker.java	13
Tagger.java	13
HeaderGrammarTagger.java	13
HeaderLengthTagger.java	16
HeaderPunctuationTagger.java	16
HeaderVaguenessTagger.java	17
HeaderVerbOrderTagger.java	20
HeaderVerbTenseTagger.java	21
MainWindow.java	22
FeedbackWindow.java	24
FeedbackList.java	25
AnalyticsWindow.java	27
Part 2 – unit tests	28
GrammarTaggerTests.java	28
LengthTaggerTests.java	29
PunctuationTaggerTests.java	30
VaguenessTaggerTests.java	31
VerbOrderTaggerTests.java	33
VerbTenseTaggerTests.java	34
HolisticTests.java	36
Appendix B – a set of Git commit messages	37
Appendix C – user guide	45

Appendix A – Code Listing

I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary.

Mateusz Przewlocki

08/04/2019

Libraries used:

- Stanford CoreNLP 3.9.2 - <https://stanfordnlp.github.io/CoreNLP/>
- Unirest for Java 1.4.9 - <http://unirest.io/java.html>
- JUnit 4.12 (for unit testing) - <https://junit.org/junit4/>

Part 1 – main code

CommitMessage.java

```
1  import edu.stanford.nlp.simple.*;
2  import java.util.ArrayList;
3  import java.util.List;
4
5  public class CommitMessage {
6      private String header;
7      private String body;
8      private String suggestedHeader = "";
9      private String suggestedBody = "";
10
11     private String sha = "";
12
13     private ArrayList<MessageTag> tags;
14     public ArrayList<String> headerTokens;
15     public ArrayList<String> headerPosTags;
16
17     public CommitMessage(String message){
18         String[] lines = message.split("\n");
19         this.header = lines[0];
20         this.suggestedHeader = header;
21
22         String body = "";
23
24         if(lines.length > 1){
25             for(int i = 1; i < lines.length; i++){
26                 body += lines[i];
27             }
28         }
29
30         tags = new ArrayList<MessageTag>();
31
32         tokeniseHeader();
33     }
34
35     public CommitMessage(String message, String sha){
36         this.sha = sha;
37
38         String[] lines = message.split("\n");
39         this.header = lines[0];
40         this.suggestedHeader = header;
41
42         String body = "";
43
44         if(lines.length > 1){
45             for(int i = 1; i < lines.length; i++){
46                 body += lines[i];
47             }
48         }
```

```
49
50         tags = new ArrayList<MessageTag>();
51
52         tokeniseHeader();
53     }
54
55     public void tokeniseHeader(){
56         Sentence headerSentence = new Sentence(header);
57         headerTokens = new
58 ArrayList<String>(headerSentence.words());
59         headerPosTags = new
60 ArrayList<String>(headerSentence.postags());
61
62         BracketsFixer.correctBrackets(headerTokens);
63     }
64
65     public String getHeader(){
66         return header;
67     }
68
69     public String getBody(){
70         return body;
71     }
72
73     public List<String> getHeaderTokens(){
74         return headerTokens;
75     }
76
77     public List<String> getHeaderPosTags(){
78         return headerPosTags;
79     }
80
81     public ArrayList<MessageTag> getTags(){
82         return tags;
83     }
84
85     public void addTag(MessageTag tag){
86         tags.add(tag);
87     }
88
89     public void printTags(){
90         System.out.println(header);
91
92         for(MessageTag tag : tags){
93             System.out.println(tag.toString());
94         }
95     }
96
97     public String getSuggestedHeader(){
98         return suggestedHeader;
99     }
```

```
98
99     public String getSuggestedBody() {
100         return suggestedBody;
101     }
102
103     public void setSuggestedHeader(String
suggestedHeader) {
104         this.suggestedHeader = suggestedHeader;
105     }
106
107     public void setSuggestedBody(String suggestedBody) {
108         this.suggestedBody = suggestedBody;
109     }
110
111     public String toString() {
112         return String.format("%s | SHA: %s", header,
sha);
113     }
114
115     public String getSha() {
116         return sha;
117     }
118
119     public void generateSuggestions() {
120         String headerSuggestion = "";
121
122         for(int i = 0; i < headerTokens.size(); i++) {
123             boolean addSpace = true;
124
125             headerSuggestion += headerTokens.get(i);
126
127             if(i + 1 < headerTokens.size()) {
128
129                 if(TokenChecker.isPunctuationToken(headerPosTags.get(i +
1)) || TokenChecker.isClosingBracket(headerTokens.get(i +
1))) {
130                     addSpace = false;
131                 }
132             }
133
134             if(TokenChecker.isBracket(headerTokens.get(i))) {
135                 addSpace = false;
136             }
137
138             if(i == headerTokens.size() - 1) {
139                 addSpace = false;
140             }
141
142             if(addSpace) {
143                 headerSuggestion += " ";
144             }
145         }
146     }
147 }
```

```
143         }
144     }
145
146     suggestedHeader = headerSuggestion;
147 }
148 }
```

MessageTag.java

```
1  import java.util.ArrayList;
2
3  public class MessageTag {
4      private String tagMessage;
5      private String suggestedChange;
6      private boolean positive;
7
8      public MessageTag(String tagMessage, boolean positive){
9          this.tagMessage = tagMessage;
10         this.positive = positive;
11     }
12
13     public MessageTag(String tagMessage, String
14         suggestedChange, boolean positive){
15         this.tagMessage = tagMessage;
16         this.suggestedChange = suggestedChange;
17         this.positive = positive;
18     }
19
20     public String getTagMessage(){
21         return this.tagMessage;
22     }
23
24     public String getSuggestedChange(){
25         return this.suggestedChange;
26     }
27
28     public boolean isPositive(){
29         return this.positive;
30     }
31
32     public String toString(){
33         String str = (positive ? "+ " : "- ");
34         str += tagMessage;
35
36         return str;
37     }
38 }
```

MessageTagger.java

```
1  import edu.stanford.nlp.simple.*;
2  import java.util.List;
3  import java.util.ArrayList;
```

```
4
5 public class MessageTagger {
6     private static int messagesAnalysed = 0;
7
8     public static void resetCounts(){
9         messagesAnalysed = 0;
10        HeaderLengthTagger.resetCount();
11        HeaderVerbOrderTagger.resetCount();
12        HeaderVerbTenseTagger.resetCount();
13        HeaderGrammarTagger.resetCount();
14        HeaderPunctuationTagger.resetCount();
15        HeaderVaguenessTagger.resetCount();
16    }
17
18    public static void generateTags(CommitMessage message){
19        messagesAnalysed++;
20        HeaderLengthTagger.tagMessage(message);
21        HeaderVerbOrderTagger.tagMessage(message);
22        HeaderVerbTenseTagger.tagMessage(message);
23        HeaderGrammarTagger.tagMessage(message);
24        HeaderPunctuationTagger.tagMessage(message);
25        HeaderVaguenessTagger.tagMessage(message);
26
27        message.generateSuggestions();
28    }
29
30    public static void main(String[] args){
31        new MainWindow();
32    }
33
34    public static int getMessagesAnalysed(){
35        return messagesAnalysed;
36    }
37 }
```

MessageGetter.java

```
1 import com.mashape.unirest.http.*;
2 import org.json.JSONArray;
3 import org.json.JSONObject;
4 import java.util.ArrayList;
5 import java.util.regex.Pattern;
6 import java.io.File;
7 import java.io.BufferedReader;
8 import java.io.FileReader;
9
10 public class MessageGetter{
11     private static String username = "mprzewlocki98";
12     private static String OAuthToken =
13         "3162fc99a81334029b8cfdcfaaba377d702a4f2b";
14     public static String GithubAPIUrl =
15         "https://api.github.com";
```

```
14
15     public static String currentRepository = "";
16
17     public static String[] parseUrl(String repositoryUrl){
18         String newUrl =
repositoryUrl.replace("https://github.com/", "");
19         String[] parsedUrl = newUrl.split("/");
20         return parsedUrl;
21     }
22
23     public static JSONObject getCommitMessageData(String
repositoryUrl, String sha){
24         String[] parsedUrl = parseUrl(repositoryUrl);
25         String username = parsedUrl[0];
26         String repo = parsedUrl[1];
27
28         JSONObject object = new JSONObject();
29
30         try{
31             HttpResponse<JsonNode> firstResponse =
Unirest.get(GithubAPIUrl + "/repos/" + username + "/" + repo +
"/commits/" + sha).basicAuth(username, OAuthToken).asJson();
32
33             object = firstResponse.getBody().getObject();
34         }catch(Exception e){
35
36         }
37
38         return object;
39     }
40
41     public static ArrayList<CommitMessage>
getCommitMessagesFromRepository(String repositoryUrl){
42         currentRepository = repositoryUrl;
43
44         ArrayList<CommitMessage> messages = new
ArrayList<CommitMessage>();
45
46         String[] parsedUrl = parseUrl(repositoryUrl);
47         String username = parsedUrl[0];
48         String repo = parsedUrl[1];
49
50         try{
51             HttpResponse<JsonNode> firstResponse =
Unirest.get(GithubAPIUrl + "/repos/" + username + "/" + repo +
"/commits?per_page=300").basicAuth(username,
OAuthToken).asJson();
52
53             JSONArray array =
firstResponse.getBody().getArray();
54
```

```
55         for(int i = 0; i < array.length(); i++){
56             String sha =
array.getJSONObject(i).getString("sha");
57             String message =
array.getJSONObject(i).getJSONObject("commit").getString("mess
age");
58             messages.add(new CommitMessage(message,
sha));
59         }
60     }catch(Exception e){
61
62     }
63
64     return messages;
65 }
66
67     public static ArrayList<CommitMessage>
getCommitMessagesFromFile(String filePath){
68         currentRepository = "";
69
70         ArrayList<CommitMessage> messages = new
ArrayList<CommitMessage>();
71
72         try{
73             File file = new File(filePath);
74             BufferedReader br = new BufferedReader(new
FileReader(file));
75             String st = br.readLine();
76
77             while(st != null){
78                 CommitMessage message = new
CommitMessage(st);
79                 messages.add(message);
80                 st = br.readLine();
81             }
82         }catch(Exception e){
83             e.printStackTrace();
84         }
85
86         return messages;
87     }
88 }
```

BracketsFixer.java

```
1  import java.util.HashMap;
2  import java.util.ArrayList;
3
4  public class BracketsFixer {
5      private static HashMap<String, String> tokensToBrackets =
new HashMap<String, String>();
6      private static boolean generatedMap = false;
```



```
7
8     public static void generateMap(){
9         tokensToBrackets.put("-LRB-", "(");
10        tokensToBrackets.put("-RRB-", ")");
11        tokensToBrackets.put("-LCB-", "{");
12        tokensToBrackets.put("-RCB-", "}");
13        tokensToBrackets.put("-LSB-", "[");
14        tokensToBrackets.put("-RSB-", "]");
15
16        generatedMap = true;
17    }
18
19    public static void correctBrackets(ArrayList<String>
tokens){
20        if(!generatedMap){
21            generateMap();
22        }
23
24        for(int i = 0; i < tokens.size(); i++){
25            String token = tokens.get(i);
26
27            if(tokensToBrackets.get(token) != null){
28                tokens.set(i,
tokensToBrackets.get(token));
29            }
30        }
31    }
32 }
```

TokenChecker.java

```
1  import java.util.HashMap;
2
3  public class TokenChecker {
4      private static HashMap<String, Boolean>
createVerbTokens() {
5          HashMap<String, Boolean> map = new HashMap<String,
Boolean>();
6          map.put("VB", true);
7          map.put("VBD", true);
8          map.put("VBG", true);
9          map.put("VBN", true);
10         map.put("VBP", true);
11         map.put("VBZ", true);
12
13         return map;
14     }
15
16     private static HashMap<String, Boolean>
createNonImperativeVerbTokens() {
17         HashMap<String, Boolean> map = new HashMap<String,
Boolean>();
```

```
18         map.put("VBD", true);
19         map.put("VBG", true);
20         map.put("VBN", true);
21         map.put("VBZ", true);
22
23         return map;
24     }
25
26     private static HashMap<String, Boolean>
27     createNounTokens() {
28         HashMap<String, Boolean> map = new HashMap<String,
29 Boolean>();
30         map.put("NN", true);
31         map.put("NNS", true);
32
33         return map;
34     }
35
36     private static HashMap<String, Boolean>
37     createProperNounTokens() {
38         HashMap<String, Boolean> map = new HashMap<String,
39 Boolean>();
40         map.put("NNP", true);
41         map.put("NNPS", true);
42
43         return map;
44     }
45
46     private static HashMap<String, Boolean>
47     createPunctuationTokens() {
48         HashMap<String, Boolean> map = new HashMap<String,
49 Boolean>();
50         map.put("#", true);
51         map.put("$", true);
52         map.put(".", true);
53         map.put(",", true);
54         map.put(":", true);
55         map.put("'", true);
56
57         return map;
58     }
59
60     private static HashMap<String, Boolean> createBrackets()
61     {
62         HashMap<String, Boolean> map = new HashMap<String,
63 Boolean>();
64         map.put("(", true);
65         map.put(")", true);
66         map.put("{", true);
67         map.put("}", true);
68         map.put("[", true);
```

```
61         map.put("]", true);
62
63         return map;
64     }
65
66     private static HashMap<String, Boolean>
67     createClosingBrackets() {
68         HashMap<String, Boolean> map = new HashMap<String,
69         Boolean>();
68         map.put(")", true);
69         map.put("}", true);
70         map.put("]", true);
71
72         return map;
73     }
74
75     private static HashMap<String, Boolean> verbTokens =
76     createVerbTokens();
77     private static HashMap<String, Boolean>
78     nonImperativeVerbTokens = createNonImperativeVerbTokens();
79     private static HashMap<String, Boolean> nounTokens =
80     createNounTokens();
81     private static HashMap<String, Boolean> properNounTokens
82     = createProperNounTokens();
83     private static HashMap<String, Boolean> punctuationTokens
84     = createPunctuationTokens();
85     private static HashMap<String, Boolean> brackets =
86     createBrackets();
87     private static HashMap<String, Boolean> closingBrackets =
88     createClosingBrackets();
89
90     public static boolean isVerbToken(String token){
91         if(verbTokens.get(token) != null){
92             return true;
93         }else{
94             return false;
95         }
96     }
97
98     public static boolean isNonImperativeVerbToken(String
99     token){
100         if(nonImperativeVerbTokens.get(token) != null){
101             return true;
102         }else{
103             return false;
104         }
105     }
106
107     public static boolean isNounToken(String token){
108         if(nounTokens.get(token) != null){
109             return true;
110         }else{
111             return false;
112         }
113     }
114 }
```

```
102         }
103
104         if(properNounTokens.get(token) != null){
105             return true;
106         }
107
108         return false;
109     }
110
111     public static boolean isProperNounToken(String
112         token) {
113         if(properNounTokens.get(token) != null){
114             return true;
115         }else{
116             return false;
117         }
118     }
119
120     public static boolean isNonProperNounToken(String
121         token) {
122         if(verbTokens.get(token) != null){
123             return true;
124         }else{
125             return false;
126         }
127     }
128
129     public static boolean isPunctuationToken(String
130         token) {
131         if(punctuationTokens.get(token) != null){
132             return true;
133         }else{
134             return false;
135         }
136     }
137
138     public static boolean isBracket(String token){
139         if(brackets.get(token) != null){
140             return true;
141         }else{
142             return false;
143         }
144     }
145
146     public static boolean isClosingBracket(String
147         token) {
148         if(closingBrackets.get(token) != null){
149             return true;
150         }else{
151             return false;
152         }
153     }
```

```
149         }
150     }
```

VagueNounChecker.java

```
1  import java.util.HashMap;
2
3  public class VagueNounChecker {
4      private static HashMap<String, Boolean> vagueNouns = new
      HashMap<String, Boolean>();
5      private static boolean generatedList = false;
6
7      public static void generateVagueNounList(){
8          vagueNouns.put("bug", true);
9          vagueNouns.put("bugs", true);
10         vagueNouns.put("feature", true);
11         vagueNouns.put("features", true);
12         vagueNouns.put("it", true);
13         vagueNouns.put("thing", true);
14         vagueNouns.put("things", true);
15         vagueNouns.put("stuff", true);
16
17         generatedList = true;
18     }
19
20     public static boolean isVagueNoun(String noun){
21         if(!generatedList){
22             generateVagueNounList();
23         }
24
25         String n = noun.toLowerCase();
26
27         return (vagueNouns.get(n) != null);
28     }
29 }
```

Tagger.java

```
1  public interface Tagger {
2      public static void tagMessage(CommitMessage message){}
3      public static int getCount(){ return 0; }
4      public static void resetCount(){}
5  }
```

HeaderGrammarTagger.java

```
1  import edu.stanford.nlp.simple.*;
2  import java.lang.Character;
3  import java.util.List;
4  import java.util.ArrayList;
5  import java.util.HashMap;
6
7  public class HeaderGrammarTagger implements Tagger {
8      private static int count = 0;
```

```
9
10     private static boolean checkIsCapitalised(String word){
11         return Character.isUpperCase(word.charAt(0));
12     }
13
14     public static String setCapitalised(String word, boolean
capitalised){
15         String w = word.toLowerCase();
16
17         if(capitalised){
18             char c =
Character.toUpperCase(word.charAt(0));
19             w = c + w.substring(1);
20         }
21
22         return w;
23     }
24
25     private static boolean
checkNonProperNounGrammar(CommitMessage message){
26         boolean incorrectGrammar = false;
27
28         List<String> tokens = message.headerTokens;
29         List<String> posTags = message.headerPosTags;
30
31         for(int i = 0; i < tokens.size(); i++){
32
33             if(!TokenChecker.isProperNounToken(posTags.get(i))){
34                 if(checkIsCapitalised(tokens.get(i))){
35                     if(i > 0){
36                         incorrectGrammar = true;
37                         message.headerTokens.set(i,
setCapitalised(message.headerTokens.get(i), false));
38                     }
39                 }else{
40                     if(i == 0){
41                         incorrectGrammar = true;
42                         message.headerTokens.set(i,
setCapitalised(message.headerTokens.get(i), true));
43                     }
44                 }
45             }
46
47             if(incorrectGrammar){
48                 MessageTag tag = new MessageTag("One or more
words in the message is incorrectly capitalised - words should
be lower case unless they are the first word in the sentence,
or proper nouns", false);
49                 message.addTag(tag);
50             }

```

```
51
52         return incorrectGrammar;
53     }
54
55     private static boolean
56     checkProperNounGrammar(CommitMessage message){
57         boolean incorrectProperNounGrammar = false;
58
59         List<String> tokens = message.headerTokens;
60         List<String> postTags = message.headerPostTags;
61
62         for(int i = 0; i < tokens.size(); i++){
63             if(TokenChecker.isProperNounToken(postTags.get(i))){
64                 if(!checkIsCapitalised(tokens.get(i))){
65                     incorrectProperNounGrammar = true;
66                     message.headerTokens.set(i,
67                     setCapitalised(message.headerTokens.get(i), true));
68                 }
69             }
70
71             if(incorrectProperNounGrammar){
72                 MessageTag tag = new MessageTag("One or more
73 proper nouns in the message is incorrectly capitalised -
74 proper nouns should always be capitalised", false);
75                 message.addTag(tag);
76             }
77             return incorrectProperNounGrammar;
78         }
79
80     public static void tagMessage(CommitMessage message){
81         boolean incorrectNounGrammar =
82         checkNonProperNounGrammar(message);
83         boolean incorrectProperNounGrammar =
84         checkProperNounGrammar(message);
85
86         if(!incorrectNounGrammar
87         && !incorrectProperNounGrammar){
88             MessageTag tag = new MessageTag("The message
89 has correct grammar", true);
90             message.addTag(tag);
91         }else{
92             count++;
93         }
94     }
95
96     public static int getCount(){
97         return count;
98     }
99 }
```

```
93     }
94
95     public static void resetCount(){
96         count = 0;
97     }
98 }
```

HeaderLengthTagger.java

```
1  public class HeaderLengthTagger implements Tagger {
2      private static int count = 0;
3
4      public static void tagMessage(CommitMessage message) {
5          int length = message.getHeader().length();
6          MessageTag tag;
7
8          if(length < 50){
9              tag = new MessageTag("The length of the header
10 is fine", true);
11          }else if(length >= 50 && length < 72){
12              tag = new MessageTag("The length of the header
13 may be too long", false);
14              count++;
15          }else{
16              tag = new MessageTag("The length of the header
17 is too long", false);
18              count++;
19          }
20
21          message.addTag(tag);
22      }
23
24      public static int getCount(){
25          return count;
26      }
27
28      public static void resetCount(){
29          count = 0;
30      }
31 }
```

HeaderPunctuationTagger.java

```
1  import edu.stanford.nlp.simple.*;
2  import java.util.List;
3
4  public class HeaderPunctuationTagger implements Tagger {
5      private static int count = 0;
6      private static String concatTokens(List<String> tokens,
7      int upTo){
8          String str = "";
9
10         for(int i = 0; i < upTo; i++){
```



```
10         str += tokens.get(i) + " ";
11     }
12
13     return str;
14 }
15
16 public static void tagMessage(CommitMessage message) {
17     List<String> tokens = message.headerTokens;
18     List<String> postTags = message.headerPostTags;
19
20     int index = postTags.size() - 1;
21
22     String lastTag = postTags.get(index);
23
24     if(TokenChecker.isPunctuationToken(lastTag)){
25         String suggestion = concatTokens(tokens,
tokens.size() - 1);
26         MessageTag tag = new MessageTag("Header should
not end with punctuation mark, as it is a title", suggestion,
false);
27         message.addTag(tag);
28
29         message.headerTokens.remove(index);
30         message.headerPostTags.remove(index);
31
32         count++;
33         return;
34     }
35 }
36
37 public static int getCount(){
38     return count;
39 }
40
41 public static void resetCount(){
42     count = 0;
43 }
44 }
```

HeaderVaguenessTagger.java

```
1  import edu.stanford.nlp.simple.*;
2  import org.json.JSONObject;
3  import java.util.List;
4  import java.util.ArrayList;
5  import java.util.HashMap;
6
7  public class HeaderVaguenessTagger implements Tagger {
8      private static int count = 0;
9
10     private static boolean checkForVerb(CommitMessage
message) {
```

```
11         for(String tag : message.headerPosTags){
12             if(TokenChecker.isVerbToken(tag)){
13                 return true;
14             }
15         }
16
17         return false;
18     }
19
20     private static boolean checkForNoun(CommitMessage
message){
21         for(String tag : message.headerPosTags){
22             if(TokenChecker.isNounToken(tag)){
23                 return true;
24             }
25         }
26
27         return false;
28     }
29
30     private static boolean checkForVagueNouns(CommitMessage
message){
31         int nouns = 0;
32         int vagueNouns = 0;
33
34         for(String tag : message.headerPosTags){
35             if(TokenChecker.isNounToken(tag)){
36                 nouns++;
37             }
38         }
39
40         for(String tag : message.headerTokens){
41             if(VagueNounChecker.isVagueNoun(tag)){
42                 vagueNouns++;
43             }
44         }
45
46         return (vagueNouns >= nouns && nouns > 0);
47     }
48
49     private static String getFileStatus(CommitMessage
message){
50
51         if(MessageGetter.currentRepository.equals("")){ return
""; }
52
53         if(message.getSha().equals("")){ return ""; }
54
55         JSONObject messageData =
MessageGetter.getCommitMessageData(MessageGetter.currentReposi
tory, message.getSha());
```

```
54         String firstFileStatus =
messageData.getJSONArray("files").getJSONObject(0).getString("
status");
55
56         return new Sentence(firstFileStatus).lemma(0);
57     }
58
59     private static String getFileName(CommitMessage message){
60
        if(MessageGetter.currentRepository.equals("")){ return
        ""; }
61         if(message.getSha().equals("")){ return ""; }
62
63         JSONObject messageData =
MessageGetter.getCommitMessageData(MessageGetter.currentReposi
tory, message.getSha());
64         String firstFile =
messageData.getJSONArray("files").getJSONObject(0).getString("
filename");
65
66         return firstFile;
67     }
68
69     public static void tagMessage(CommitMessage message){
70         boolean hasVerb = checkForVerb(message);
71         boolean hasNoun = checkForNoun(message);
72         boolean hasOnlyVagueNouns =
checkForVagueNouns(message);
73
74         if(!hasVerb && !hasNoun){
75             MessageTag tag = new MessageTag("Message is
too vague: does not describe what change was made at all",
"add a verb and a noun", false);
76             message.addTag(tag);
77
78             count++;
79         }else if(!hasVerb){
80             MessageTag tag = new MessageTag("Message is
too vague: does not contain any meaningful verbs", "add a
verb", false);
81             message.addTag(tag);
82
83             if(hasOnlyVagueNouns){
84                 MessageTag tag2 = new MessageTag("Message
is too vague: all nouns in the message are considered vague.
Consider using more descriptive nouns", "add a more
descriptive noun", false);
85                 message.addTag(tag2);
86             }
87
88             count++;
```

```
89         }else if(!hasNoun){
90             MessageTag tag = new MessageTag("Message is
too vague: does not contain any meaningful nouns", "add a
noun", false);
91             message.addTag(tag);
92
93             count++;
94         }else if(hasOnlyVagueNouns){
95             MessageTag tag = new MessageTag("Message is
too vague: all nouns in the message are considered vague.
Consider using more descriptive nouns", "add a more
descriptive noun", false);
96             message.addTag(tag);
97
98             count++;
99         }
100     }
101
102     public static int getCount(){
103         return count;
104     }
105
106     public static void resetCount(){
107         count = 0;
108     }
109 }
```

HeaderVerbOrderTagger.java

```
1  import edu.stanford.nlp.simple.*;
2  import java.util.List;
3  import java.util.ArrayList;
4  import java.util.HashMap;
5
6  public class HeaderVerbOrderTagger implements Tagger {
7      private static int count = 0;
8
9      public static void tagMessage(CommitMessage message){
10         boolean hasIncorrectOrder = false;
11         int firstVerbIndex = -1;
12
13         ArrayList<String> tokens = message.headerTokens;
14         ArrayList<String> postTags = message.headerPostTags;
15
16         if(!TokenChecker.isVerbToken(postTags.get(0))){
17             hasIncorrectOrder = true;
18         }
19
20         for(int i = 0; i < postTags.size(); i++){
21             if(TokenChecker.isVerbToken(postTags.get(i))){
22                 firstVerbIndex = i;
23                 break;
24             }
25         }
26     }
27 }
```

```
24         }
25     }
26
27     if(hasIncorrectOrder){
28         MessageTag tag = new MessageTag("The message
should start with a verb.", false);
29         message.addTag(tag);
30
31         if(firstVerbIndex > 0){
32             String firstVerb =
tokens.get(firstVerbIndex);
33             String firstVerbPosTag =
posTags.get(firstVerbIndex);
34
35             tokens.remove(firstVerbIndex);
36             posTags.remove(firstVerbIndex);
37
38             tokens.add(0, firstVerb);
39             posTags.add(0, firstVerbPosTag);
40         }
41
42         count++;
43     }
44 }
45
46 public static int getCount(){
47     return count;
48 }
49
50 public static void resetCount(){
51     count = 0;
52 }
53 }
```

HeaderVerbTenseTagger.java

```
1  import edu.stanford.nlp.simple.*;
2  import java.util.List;
3  import java.util.ArrayList;
4  import java.util.HashMap;
5
6  public class HeaderVerbTenseTagger implements Tagger {
7      private static int count = 0;
8
9      private static String getBaseForm(String word){
10         String w = new Sentence(word).lemma(0);
11
12         return w;
13     }
14
15     public static void tagMessage(CommitMessage message){
16         boolean hasVerbs = false;
```

```
17         boolean hasIncorrectTense = false;
18         boolean chcekTense = true;
19
20         List<String> tokens = message.headerTokens;
21         List<String> postTags = message.headerPostTags;
22
23         for(int i = 0; i < tokens.size(); i++){
24             String tag = postTags.get(i);
25
26             if(tag.equals("RP") || tag.equals("IN")){
27                 checkTense = false;
28             }
29
30             if(TokenChecker.isVerbToken(tag) &&
checkTense){
31                 hasVerbs = true;
32             }
33
34             if(TokenChecker.isNonImperativeVerbToken(tag)
&& checkTense){
35                 hasIncorrectTense = true;
36                 message.headerTokens.set(i,
getBaseForm(message.headerTokens.get(i)));
37             }
38         }
39
40         if(hasIncorrectTense){
41             MessageTag tag = new MessageTag("Verbs should
be in the present imperative form", false);
42             message.addTag(tag);
43
44             count++;
45         }else if(hasVerbs){
46             MessageTag tag = new MessageTag("All verbs are
in the present imperative form.", true);
47             message.addTag(tag);
48         }
49     }
50
51     public static int getCount(){
52         return count;
53     }
54
55     public static void resetCount(){
56         count = 0;
57     }
58 }
```

MainWindow.java

```
1 import javax.swing.*;
2 import java.awt.event.*;
```

```
3  import java.awt.Dimension;
4  import java.util.ArrayList;
5
6  public class MainWindow {
7      public MainWindow () {
8          JFrame frame = new JFrame();
9          frame.setSize(500, 300);
10         frame.setMinimumSize(new Dimension(500, 300));
11
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         JPanel panel = new JPanel();
15         BorderLayout layout = new BorderLayout(panel,
16         BorderLayout.Y_AXIS);
17         panel.setLayout(layout);
18
19         JLabel welcome = new JLabel("Welcome to Git commit
20         message analysis through natural language processing");
21         JLabel inputMessage = new JLabel("Please input a
22         commit message for analysis:");
23         JLabel inputRepo = new JLabel("or type the web
24         address of a repository:");
25         JLabel inputFile = new JLabel("or type the address
26         of a file:");
27
28         JTextField singleMessageTextField = new
29         JTextField();
30         JButton singleMessageButton = new JButton("Analyse
31         single message");
32
33         singleMessageButton.addActionListener(new
34         ActionListener() {
35             @Override
36             public void actionPerformed(ActionEvent e) {
37                 CommitMessage message = new
38                 CommitMessage(singleMessageTextField.getText());
39                 MessageTagger.generateTags(message);
40                 new FeedbackWindow(message);
41             }
42         });
43
44         JTextField repoTextField = new JTextField();
45         JButton repoButton = new JButton("Analyse
46         repository");
47
48         repoButton.addActionListener(new ActionListener() {
49             @Override
50             public void actionPerformed(ActionEvent e) {
51                 MessageTagger.resetCounts();
52             }
53         });
54     }
55 }
```

```
42         ArrayList<CommitMessage> messages =
    MessageGetter.getCommitMessagesFromRepository(repoTextField.ge
    tText());
43
44         messages.parallelStream().forEach(message
    -> MessageTagger.generateTags(message));
45         new FeedbackList(messages);
46     }
47     });
48
49     JTextField fileTextField = new JTextField();
50     JButton fileButton = new JButton("Analyse file");
51
52     fileButton.addActionListener(new ActionListener() {
53         @Override
54         public void actionPerformed(ActionEvent e) {
55             MessageTagger.resetCounts();
56
57             ArrayList<CommitMessage> messages =
    MessageGetter.getCommitMessagesFromFile(fileTextField.getText(
    ));
58
59             messages.parallelStream().forEach(message
    -> MessageTagger.generateTags(message));
60             new FeedbackList(messages);
61         }
62     });
63
64     panel.add(welcome);
65     panel.add(inputMessage);
66     panel.add(singleMessageTextField);
67     panel.add(singleMessageButton);
68     panel.add(inputRepo);
69     panel.add(repoTextField);
70     panel.add(repoButton);
71     panel.add(inputFile);
72     panel.add(fileTextField);
73     panel.add(fileButton);
74
75     frame.add(panel);
76     frame.setVisible(true);
77 }
78 }
```

FeedbackWindow.java

```
1 import javax.swing.*;
2 import java.awt.*;
3 import java.util.ArrayList;
4
5 public class FeedbackWindow {
```



```
6         public static ArrayList<String>
getTagsAsString(CommitMessage message) {
7             ArrayList<String> list = new ArrayList<String>();
8
9             for(MessageTag tag : message.getTags()){
10                 list.add(tag.toString());
11             }
12
13             return list;
14         }
15
16         public FeedbackWindow(CommitMessage message) {
17             JFrame frame = new JFrame("Commit message
feedback");
18             frame.setSize(500, 200);
19
20             JPanel panel = new JPanel(new BorderLayout());
21             frame.add(panel);
22
23             JLabel label = new JLabel(message.getHeader());
24             panel.add(label, BorderLayout.PAGE_START);
25
26             DefaultListModel<String> listModel = new
DefaultListModel<String>();
27             ArrayList<String> tagStringList =
getTagsAsString(message);
28
29             for(String s : tagStringList){
30                 listModel.addElement(s);
31             }
32
33             JList<String> list = new JList<String>(listModel);
34             panel.add(new JScrollPane(list),
BorderLayout.CENTER);
35
36             JLabel suggestedChange = new JLabel("Suggested new
message: " + message.getSuggestedHeader());
37             panel.add(suggestedChange, BorderLayout.PAGE_END);
38
39             frame.setVisible(true);
40         }
41 }
```

FeedbackList.java

```
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.*;
4 import java.util.ArrayList;
5
6 public class FeedbackList {
```

```
7      public FeedbackList(ArrayList<CommitMessage> messages) {
8          JFrame frame = new JFrame("Repository feedback");
9          frame.setSize(500, 800);
10
11          JPanel panel = new JPanel(new BorderLayout());
12          frame.add(panel);
13
14          JLabel label = new JLabel("Double click on a message
to see the feedback for that message");
15          panel.add(label, BorderLayout.PAGE_START);
16
17          DefaultListModel<CommitMessage> listModel = new
DefaultListModel<CommitMessage>();
18
19          for(CommitMessage message : messages){
20              listModel.addElement(message);
21          }
22
23          JList<CommitMessage> list = new
JList<CommitMessage>(listModel);
24
25          list.addMouseListener(new MouseAdapter() {
26              public void mouseClicked(MouseEvent evt){
27                  if(evt.getClickCount() == 2){
28                      int index =
list.locationToIndex(evt.getPoint());
29                      CommitMessage message =
listModel.getElementAt(index);
30
31                      new FeedbackWindow(message);
32                  }
33              }
34          });
35
36          panel.add(new JScrollPane(list),
BorderLayout.CENTER);
37
38          JButton analyticsButton = new JButton("Analytics");
39
40          analyticsButton.addActionListener(new
ActionListener() {
41              @Override
42              public void actionPerformed(ActionEvent e){
43                  new AnalyticsWindow();
44              }
45          });
46
47          panel.add(analyticsButton, BorderLayout.SOUTH);
48
49          frame.setVisible(true);
```

```
50     }  
51 }
```

AnalyticsWindow.java

```
1  import javax.swing.*;  
2  import java.awt.*;  
3  import java.util.ArrayList;  
4  
5  public class AnalyticsWindow {  
6      public AnalyticsWindow() {  
7          JFrame frame = new JFrame("Analytics");  
8          frame.setSize(600, 200);  
9  
10         JPanel panel = new JPanel();  
11         BoxLayout layout = new BoxLayout(panel,  
12         BoxLayout.Y_AXIS);  
13         panel.setLayout(layout);  
14         JLabel messagesAnalysed = new JLabel("Messages  
15         analysed: " + MessageTagger.getMessagesAnalysed());  
16         JLabel headerData = new JLabel("- Message header  
17         data -");  
18         JLabel lengthTags = new JLabel("Messages with  
19         incorrect header length: " + HeaderLengthTagger.getCount());  
20         JLabel grammarTags = new JLabel("Messages with  
21         incorrect grammar: " + HeaderGrammarTagger.getCount());  
22         JLabel punctuationTags = new JLabel("Messages with  
23         incorrect punctuation: " +  
24         HeaderPunctuationTagger.getCount());  
25         JLabel verbTenseTags = new JLabel("Messages with  
26         incorrect verb tense: " + HeaderVerbTenseTagger.getCount());  
27         JLabel verbOrderTags = new JLabel("Messages with  
28         incorrect verb order: " + HeaderVerbOrderTagger.getCount());  
29         JLabel vagueTags = new JLabel("Messages that are  
30         considered vague: " + HeaderVaguenessTagger.getCount());  
31  
32         panel.add(messagesAnalysed);  
33         panel.add(headerData);  
34         panel.add(lengthTags);  
35         panel.add(grammarTags);  
36         panel.add(punctuationTags);  
37         panel.add(verbTenseTags);  
38         panel.add(verbOrderTags);  
39         panel.add(vagueTags);  
40  
41         frame.add(panel);  
42  
43         frame.setVisible(true);  
44     }  
45 }
```

Part 2 – unit tests

GrammarTaggerTests.java

```
1  import org.junit.*;
2  import static org.junit.Assert.assertTrue;
3
4  public class GrammarTaggerTests {
5      @Test
6      public void TestGrammarCorrect(){
7          CommitMessage message = new CommitMessage("This has
correct grammar");
8          String tagMessage = "The message has correct
grammar";
9
10         HeaderGrammarTagger.tagMessage(message);
11
12         boolean hasTag = false;
13
14         for(MessageTag tag : message.getTags()){
15             if(tag.getTagMessage().equals(tagMessage)){
16                 hasTag = true;
17             }
18         }
19
20         assertTrue(hasTag);
21     }
22
23     @Test
24     public void TestGrammarIncorrectNoun(){
25         CommitMessage message = new CommitMessage("This has
incorrect Grammar");
26         String tagMessage = "One or more words in the
message is incorrectly capitalised - words should be lower
case unless they are the first word in the sentence, or proper
nouns";
27
28         HeaderGrammarTagger.tagMessage(message);
29
30         boolean hasTag = false;
31
32         for(MessageTag tag : message.getTags()){
33             if(tag.getTagMessage().equals(tagMessage)){
34                 hasTag = true;
35             }
36         }
37
38         assertTrue(hasTag);
39     }
40
41     @Test
42     public void TestGrammarIncorrectProperNoun(){
```

```
43         CommitMessage message = new CommitMessage("Fix
london");
44         String tagMessage = "One or more proper nouns in
the message is incorrectly capitalised - proper nouns should
always be capitalised";
45
46         HeaderGrammarTagger.tagMessage(message);
47
48         boolean hasTag = false;
49
50         for(MessageTag tag : message.getTags()){
51             if(tag.getTagMessage().equals(tagMessage)){
52                 hasTag = true;
53             }
54         }
55
56         assertTrue(hasTag);
57     }
58 }
```

LengthTaggerTests.java

```
1  import org.junit.*;
2  import static org.junit.Assert.assertTrue;
3
4  public class LengthTaggerTests {
5      @Test
6      public void TestHeaderLengthTaggerOK(){
7          CommitMessage OKMessage = new CommitMessage("This
message is below 50 characters");
8          String lengthMessage = "The length of the header is
fine";
9
10         HeaderLengthTagger.tagMessage(OKMessage);
11
12         boolean hasTag = false;
13
14         for(MessageTag tag : OKMessage.getTags()){
15             if(tag.getTagMessage().equals(lengthMessage)){
16                 hasTag = true;
17             }
18         }
19
20         assertTrue(hasTag);
21     }
22
23     @Test
24     public void TestHeaderLengthTaggerDubious(){
25         CommitMessage dubiousMessage = new
CommitMessage("This message is over 50 characters but not over
72");
```

```
26         String lengthMessage = "The length of the header may
    be too long";
27
28         HeaderLengthTagger.tagMessage(dubiousMessage);
29
30         boolean hasTag = false;
31
32         for(MessageTag tag : dubiousMessage.getTags()){
33             if(tag.getTagMessage().equals(lengthMessage)){
34                 hasTag = true;
35             }
36         }
37
38         assertTrue(hasTag);
39     }
40
41     @Test
42
43     public void TestHeaderLengthTaggerTooLong(){
44         CommitMessage tooLongMessage = new
    CommitMessage("This message is definitely far too long and
    should be tagged accordingly.");
45         String lengthMessage = "The length of the header is
    too long";
46
47         HeaderLengthTagger.tagMessage(tooLongMessage);
48
49         boolean hasTag = false;
50
51         for(MessageTag tag : tooLongMessage.getTags()){
52             if(tag.getTagMessage().equals(lengthMessage)){
53                 hasTag = true;
54             }
55         }
56
57         assertTrue(hasTag);
58     }
59 }
```

PunctuationTaggerTests.java

```
1  import org.junit.*;
2  import static org.junit.Assert.assertTrue;
3
4  public class PunctuationTaggerTests {
5      @Test
6      public void
    TestPunctuationTaggerFindssFinalPunctuation(){
7          CommitMessage message = new CommitMessage("There
    should be no punctuation.");
8          String tagMessage = "Header should not end with
    punctuation mark, as it is a title";
```

```
9
10         HeaderPunctuationTagger.tagMessage(message);
11
12         boolean hasTag = false;
13
14         for(MessageTag tag : message.getTags()){
15             if(tag.getTagMessage().equals(tagMessage)){
16                 hasTag = true;
17             }
18         }
19
20         assertTrue(hasTag);
21     }
22 }
```

VaguenessTaggerTests.java

```
1  import org.junit.*;
2  import static org.junit.Assert.assertTrue;
3  import static org.junit.Assert.assertFalse;
4
5  public class VaguenessTaggerTests {
6      @Test
7      public void TestNonVagueMessage(){
8          CommitMessage message = new CommitMessage("Add
9  Polish translation");
10         String tagMessage1 = "Message is too vague: does
11         not describe what change was made at all";
12         String tagMessage2 = "Message is too vague: does not
13         contain any meaningful verbs";
14         String tagMessage3 = "Message is too vague: all
15         nouns in the message are considered vague. Consider using more
16         descriptive nouns";
17         String tagMessage4 = "Message is too vague: does not
18         contain any meaningful nouns";
19
20         HeaderVaguenessTagger.tagMessage(message);
21
22         boolean hasTag = false;
23
24         for(MessageTag tag : message.getTags()){
25             if(tag.getTagMessage().equals(tagMessage1)){
26                 hasTag = true;
27             }
28
29             if(tag.getTagMessage().equals(tagMessage2)){
30                 hasTag = true;
31             }
32
33             if(tag.getTagMessage().equals(tagMessage3)){
34                 hasTag = true;
35             }
36
37             if(tag.getTagMessage().equals(tagMessage4)){
38                 hasTag = true;
39             }
40         }
41     }
42 }
```

```
30
31         if(tag.getTagMessage().equals(tagMessage4)){
32             hasTag = true;
33         }
34     }
35
36     assertFalse(hasTag);
37 }
38
39 @Test
40 public void TestOverlyVagueMessage(){
41     CommitMessage message = new CommitMessage("0");
42     String tagMessage = "Message is too vague: does not
describe what change was made at all";
43
44     HeaderVaguenessTagger.tagMessage(message);
45
46     boolean hasTag = false;
47
48     for(MessageTag tag : message.getTags()){
49         if(tag.getTagMessage().equals(tagMessage)){
50             hasTag = true;
51         }
52     }
53
54     assertTrue(hasTag);
55 }
56
57 @Test
58 public void TestVagueNoVerbMessage(){
59     CommitMessage message = new CommitMessage("Cosmetic
improvements");
60     String tagMessage = "Message is too vague: does not
contain any meaningful verbs";
61
62     HeaderVaguenessTagger.tagMessage(message);
63
64     boolean hasTag = false;
65
66     for(MessageTag tag : message.getTags()){
67         if(tag.getTagMessage().equals(tagMessage)){
68             hasTag = true;
69         }
70     }
71
72     assertTrue(hasTag);
73 }
74
75 @Test
76 public void TestVagueNoNounMessage(){
77     CommitMessage message = new CommitMessage("Added");
```



```
78         String tagMessage = "Message is too vague: does not
    contain any meaningful nouns";
79
80         HeaderVaguenessTagger.tagMessage(message);
81
82         boolean hasTag = false;
83
84         for(MessageTag tag : message.getTags()){
85             if(tag.getTagMessage().equals(tagMessage)){
86                 hasTag = true;
87             }
88         }
89
90         assertTrue(hasTag);
91     }
92
93     @Test
94     public void TestVagueNounsMessage(){
95         CommitMessage message = new CommitMessage("Fix
    bug");
96         String tagMessage = "Message is too vague: all
    nouns in the message are considered vague. Consider using more
    descriptive nouns";
97
98         HeaderVaguenessTagger.tagMessage(message);
99
100        boolean hasTag = false;
101
102        for(MessageTag tag : message.getTags()){
103            if(tag.getTagMessage().equals(tagMessage)){
104                hasTag = true;
105            }
106        }
107
108        assertTrue(hasTag);
109    }
110 }
```

VerbOrderTaggerTests.java

```
1  import org.junit.*;
2  import static org.junit.Assert.assertTrue;
3  import static org.junit.Assert.assertFalse;
4  import static org.junit.Assert.assertEquals;
5
6  public class VerbOrderTaggerTests {
7      @Test
8      public void TestFindsCorrectOrder(){
9          CommitMessage message = new CommitMessage("remove
    bug");
```

```
10         String tagMessage = "The message should start with a
    verb.";
11
12         HeaderVerbOrderTagger.tagMessage(message);
13
14         boolean hasTag = false;
15
16         for(MessageTag tag : message.getTags()){
17             if(tag.getTagMessage().equals(tagMessage)){
18                 hasTag = true;
19             }
20         }
21
22         assertFalse(hasTag);
23     }
24
25     @Test
26     public void TestFindsIncorrectOrder(){
27         CommitMessage message = new CommitMessage("bug
    removed");
28         String tagMessage = "The message should start with a
    verb.";
29
30         HeaderVerbOrderTagger.tagMessage(message);
31
32         boolean hasTag = false;
33
34         for(MessageTag tag : message.getTags()){
35             if(tag.getTagMessage().equals(tagMessage)){
36                 hasTag = true;
37             }
38         }
39
40         assertTrue(hasTag);
41     }
42
43     @Test
44     public void TestChangesIncorrectOrder(){
45         CommitMessage message = new CommitMessage("bug
    removed");
46         HeaderVerbOrderTagger.tagMessage(message);
47         message.generateSuggestions();
48
49         assertEquals(message.getSuggestedHeader(), "removed
    bug");
50     }
51 }
```

VerbTenseTaggerTests.java

```
1 import org.junit.*;
2 import static org.junit.Assert.assertTrue;
```

```
3  import static org.junit.Assert.assertEquals;
4
5  public class VerbTenseTaggerTests {
6      @Test
7      public void TestFindsCorrectTense(){
8          CommitMessage message = new CommitMessage("Fix
9  bug");
10         String tagMessage = "All verbs are in the present
11         imperative form.";
12
13         HeaderVerbTenseTagger.tagMessage(message);
14
15         boolean hasTag = false;
16
17         for(MessageTag tag : message.getTags()){
18             if(tag.getTagMessage().equals(tagMessage)){
19                 hasTag = true;
20             }
21         }
22
23         assertTrue(hasTag);
24     }
25
26     @Test
27     public void TestFindsIncorrectTense(){
28         CommitMessage message = new CommitMessage("Fixed
29         bug");
30         String tagMessage = "Verbs should be in the present
31         imperative form";
32
33         HeaderVerbTenseTagger.tagMessage(message);
34
35         boolean hasTag = false;
36
37         for(MessageTag tag : message.getTags()){
38             if(tag.getTagMessage().equals(tagMessage)){
39                 hasTag = true;
40             }
41         }
42
43         assertTrue(hasTag);
44     }
45
46     @Test
47     public void TestChangesIncorrectTense(){
48         CommitMessage message = new CommitMessage("Fixed
49         bug");
50         HeaderVerbTenseTagger.tagMessage(message);
51         message.generateSuggestions();
52     }
53 }
```

```
48         assertEquals(message.getSuggestedHeader(), "Fix
    bug");
49     }
50 }
```

HolisticTests.java

```
1  import org.junit.*;
2  import static org.junit.Assert.assertTrue;
3  import static org.junit.Assert.assertEquals;
4
5  public class HolisticTests {
6      @Test
7      public void TestMessage1(){
8          CommitMessage message = new CommitMessage("fix
    bug.");
9          MessageTagger.generateTags(message);
10         message.generateSuggestions();
11         assertEquals(message.getSuggestedHeader(), "Fix
    bug");
12     }
13
14     @Test
15     public void TestMessage2(){
16         CommitMessage message = new
    CommitMessage("translation added");
17         MessageTagger.generateTags(message);
18         message.generateSuggestions();
19         assertEquals(message.getSuggestedHeader(), "Add
    translation");
20     }
21
22     @Test
23     public void TestMessage3(){
24         CommitMessage message = new CommitMessage("added
    music, added game, removed feature");
25         MessageTagger.generateTags(message);
26         message.generateSuggestions();
27         assertEquals(message.getSuggestedHeader(), "Add
    music, add game, remove feature");
28     }
29 }
```

Appendix B – a set of Git commit messages

Below is a selection of 201 commit messages from Git repositories, along with the link to the repository they were found in and the date of access. This is also included with the code under the file name “messages.txt”.

<https://github.com/mprzewlocki98/second-year-major-project/> -
Accessed 14/11/2018

Add screenshots for readme
adding back music into main menu
updated music to non-copyright
changed aspect ratio from free aspect to 16:9
correcting some of the game's settings
updating readme file
cleaning up folder structure
Fix tests
optimise metalGame
Cosmetic improvements

<https://github.com/google-research/bert> - Accessed 14/11/2018

fix test
Running through pyformat to meet Google code standards
return API method "convert_tokens_to_ids(vocab, tokens)" and add
"con...
Fixing comment
Fixing bug introducing in classification
Add method for converting ids to tokens.
Update README to clarify feature extraction
Updating chainer FAQ
Fixing CoLA predict mode
Fixing python2 logging for extract_features.py

<https://github.com/facebook/react> - Accessed 14/11/2018

Add 16.6.3 Changelog (#14223)
Update error codes
Save CI-built node_modules as build artifacts (#14205)

Add regression test for #14188 (#14197)
Update Readme (#14176)
Simplify CSS shorthand property warning (#14183)
fix typo
Use unique thread ID for each partial render to access Context (#14182)
SimpleMemoComponent should warn if a ref is given (#14178)
Warn about conflicting style values during updates (#14181)

<https://github.com/TheAlgorithms/Python> - Accessed 14/11/2018

Added axes label to the plot
Update absMin.py
Update FindMin.py (#601)
fix division by float issue in range heap.py
Minor changes to README.md (#599)
Re-design psnr.py code and change image names (#592)
Update max_sub_array.py (#597)
Update 3n+1.py
Added b16, b32, a85, abs, absMax, absMin
Update singly_linked_list.py

<https://github.com/jantic/DeOldify> - Accessed 14/11/2018

One more fix/hack then I think this Colab pillow nightmare is over!
Pillow pillow pillow.
Another attempt at pillow fix on colab notebook
Typo
Another attempt to fix colab pil issue (wat wat wat)
Using Colab friendly array to image conversion
Attempting pil fix for colab notebook again
Attempting to fix pillow issue
Merging in updated colab notebook that can now read test images and w...
colab notebook updated to work with local Drive images

<https://github.com/30-seconds/30-seconds-of-code> - Accessed 14/11/2018

added edge cases for everyNth test

Fix module generation

Coverage cleanup

Fixed coverage for Codacy

Fix coverage

Fix tdd

Update tdd

1.2.2

Minor fixes

Fix package

<https://github.com/s0md3v/XSSStrike> - Accessed 14/11/2018

warn user when no parameters are supplied (#119)

Proxy Support (Resolves #55)

added a question about blind xss

Blind XSS Support

updated changelog

changed version number in banner

added "payload encoding" to features

Ability to encode payloads, Fixed a bug in bruteforcer

Verbose switch, Fixes #71, Fixes #93

potential fix for #93

<https://github.com/enquirer/enquirer> - Accessed 14/11/2018

2.0.7

render pointer with validation message

allow returning false in validation

ci: test on Linux, macOS and Windows

ci: test Node.js 8, 10 and 11

Update .verb.md

support appveyor

fix example

adds multiscale gif
fix table

<https://github.com/GoogleChromeLabs/squoosh> - Accessed 14/11/2018

Allow text fields next to range inputs be empty (yeah that's horrendo...

Avoid "update found" on initial load.

Fix typos [emoji]

Adding readme, privacy section, reducing resolution of analytics data.

Resetting pinch zoom (#261)

Not entirely sure why this causes dev to fail, but this fixes it.

Preload test (#262)

I'm calling this 1.0

Adding manifest to headers

Removing old file from serviceworker

<https://github.com/leonardomso/33-js-concepts> - Accessed 14/11/2018

Polish translation added

Polish translation ready

Added link to Polish translation

new Pure Functions article

new Object.assign article

new Closures article

Add new DOM article

a new video for data structures

Add article about Design Pattern from Medium.

How to traverse DOM

<https://github.com/flutter/flutter> - Accessed 10/12/2018

Fix semantics compiler for offstage children (#24862)

Add animations to SliverAppBar doc (#25091)

Fix typo in documentation (#25003)

Add a flutter-attach entry point for fuchsia (#24878)

Clarify dart:ui dependencies in foundation library (#24868)

Smoke test building IPA and APK on supported platforms (#24601)
Remove superfluous "install" (#24836)
Updating readme templates for newly created projects (#24725)
Roll engine be973ea1961..72c7a756722 (#24828)
Merge analaytics from docs site and flutter.io site. (#24825)

<https://github.com/wagoodman/dive/> - Accessed 10/12/2018

version v0.5.0 bump
handle scratch images (closes #76)
Export metrics to a file (#122)
Performance tweaks (#127)
Refactor image preprocessing (#121)
bump patch version in docs
read entire json file on image parsing
added mac download link; formatting
formatting windows title
archive override for windows (zip)

<https://github.com/satwikkansal/wtfpython> - Accessed 10/12/2018

Rephrase sentence to make more sense (#109)
fix typo (#107)
match order of display to if/else order (#105)
make the function *call* more visible (#104)
Update broken link
Fixed input-output of 4th example explanation code (#92)
Fix Python 3 compatibility in the irrelevant code!
Fix an error in the irrelevant code!
Fix incorrect output in "Mysterious key type conversion" example
Simplify explanation

<https://github.com/Eugeniy/terminus/> - Accessed 10/12/2018

build fix
Revert "bumped webpack"
potential fix for xterm double-paste (#468)

```
xterm copy-on-select (fixes #400)
make scroll-on-input behaviour configurable (fixes #543)
bumped webpack
fixed settings sidebar offset (fixes #549)
hotkey fixes
nicer scrollbars (fixes #440)
don't crash if no global spawn hotkey is assigned (#540)
```

<https://github.com/wasmerio/wasmer/> - Accessed 10/12/2018

Fix clippy warnings returning the result of a let binding from a block

Fix clippy warnings manual implementation of an assign operation

Fix clippy warnings unused imports

Fix clippy unused variable: `instance`

Fix unneeded return statement

Fix clippy long literal lacking separators warnings

Fix redundant_field_names

Improved README styling

Improved docs with installation instructions

Updated version to 0.1.3

<https://github.com/pcottle/learnGitBranching> - Accessed 10/12/2018

fix a little typo in cn

Use Object.values instead of _.each : 96ddb50

Create simple debounce and throttle instead of _.

Add yarn.lock

Remove module unuse

Change npm to yarn

Add some info about this repository

Restore `src/levels` in `Gruntfile.js`

remove forgot in 'no more native' a9dd27c

Avoid display 900px in small screen

<https://github.com/oussamahamdaoui/forgJs> - Accessed 10/12/2018

better code smell

refactoring

add code climate badge

add boolean type and fix code coverege

add oneOf

update readme

adding url type

password and user thype

README.md typo fixes [ci skip]

adding password type

<https://github.com/dotnet/winforms> - Accessed 10/12/2018

update issue-guide.md to include new PR labels (#203)

disable test signing while arcade is looking into issue (#220)

Update dependency files (#211)

disable test signing for now (#206)

Remove SYTEM_WEB, UNUSED, SOAP_FORMATTER and commented code from
Syst...

Suppress SecurityCritical attributes (#201)

Use ValueTuple instead of Tuple for internal cache

Move ApiHelper to shared code and add tests

Clean out Windows 9x code (#154)

Update dependencies from <https://github.com/dotnet/arcade> build 650
(#...

Appendix C – user guide

Prerequisites for running the application

- Java 8
- Gradle

Installation instructions

1. Place the folder with the application in any location, then navigate to that location in Terminal (Unix-based systems) or Command Prompt (Windows)
2. Type “gradle build” and wait for it to finish building the project. This is a necessary step as it will download the Stanford CoreNLP .jar and Unirest .jar files, which are too large to include in this archive.
3. Type “gradle run” after the project is built to run the program.

Running the application

When the application is first run, an interface like the following will appear on the screen.

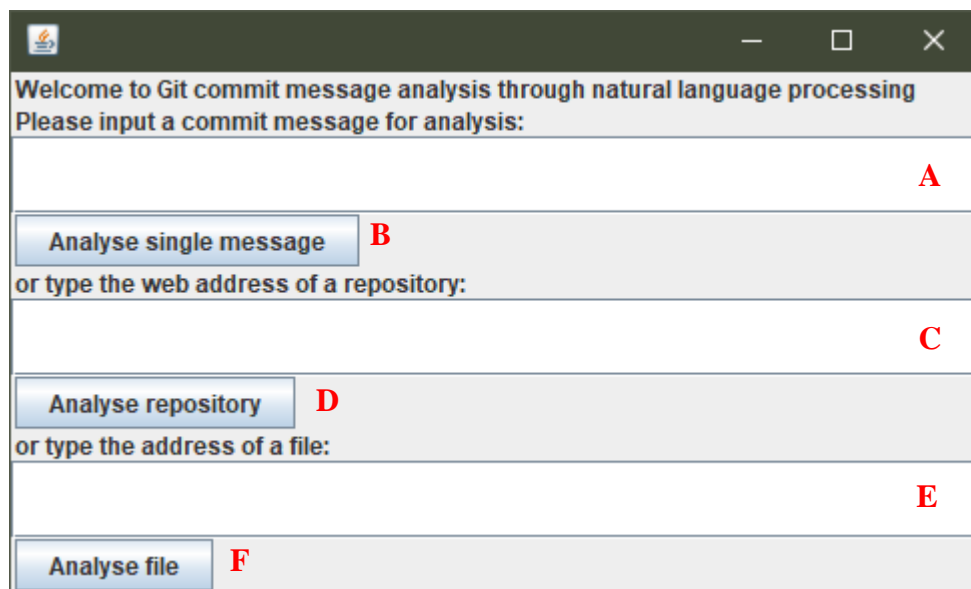


Figure C.1 – initial application interface

- | | |
|---|---|
| A) Text field for analysing single commit message | E) Text field for analysing a .txt file |
| B) Button for analysing single message | F) Button for analysing a text file |
| C) Text field for analysing a GitHub repository | |
| D) Button for analysing a repository | |

To analyse a single commit message, type a commit message in field A then press button B. An interface like the following should appear on the screen.

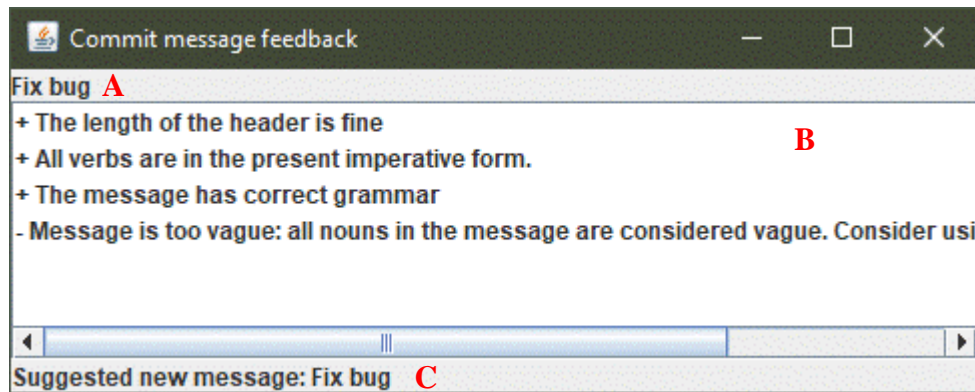


Figure C.2 – feedback for a single commit message

- A) Original commit message
- B) List of feedback items
- C) Message suggestion

Note the “+” and “-” symbols present next to each item of feedback in list B. These show whether the item of feedback is positive or negative and are insights into areas of improvement for the commit message. Note also the message suggestion C which provides a suggestion on how the message can be improved, if the program can find one.

To analyse a GitHub repository, type a web address in the form “https://github.com/username/repository” into field C, then click button D. An interface like the following should then appear on the screen.

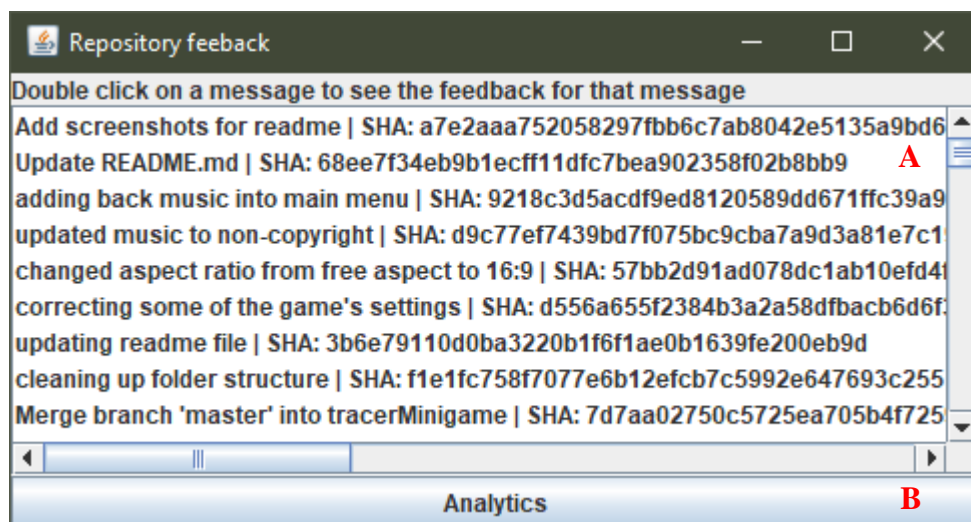


Figure C.3 – feedback for a repository

- A) List of commit messages
- B) Button for analytics

To view feedback for a commit message, simply double-click the message in list A. A window like the one in Figure C.2 should show on the screen with all applicable feedback. To view analytical data about the analysis of the messages in the repository, click button B. A window like the following should then appear on the screen.

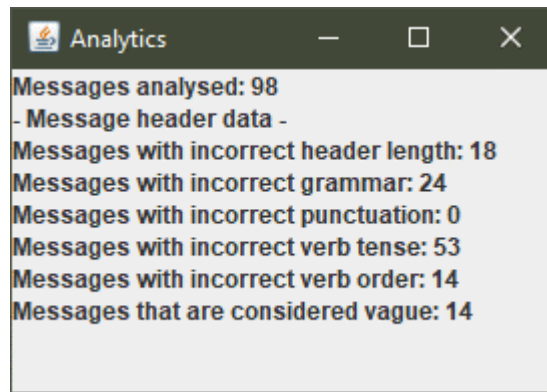


Figure C.3 – analytics window

You can then close this window once you are done viewing it.

To analyse a text file, type the exact address of a .txt file in field E, then click button F. A window like the one in Figure C.3 should appear on the screen. This window operates exactly in the same way as for analysing a GitHub repository.

When you are done using the application, you may simply close all windows the application has created.