Git Commit Message Analysis – Appendices

Contents

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

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

public class CommitMessage {
	private String header;
	private String body;
	private String suggestedHeader = "";
	private String suggestedBody = "";

	private String sha = "";

	private ArrayList<MessageTag> tags;
	public ArrayList<String> headerTokens;
	public ArrayList<String> headerPosTags;

	public CommitMessage(String message){
		String[] lines = message.split("\n");
		this.header = lines[0];
		this.suggestedHeader = header;

		String body = "";

		if(lines.length > 1){
			for(int i = 1; i < lines.length; i++){
				body += lines[i];
			}
		}

		tags = new ArrayList<MessageTag>();

		tokeniseHeader();
	}

	public CommitMessage(String message, String sha){
		this.sha = sha;

		String[] lines = message.split("\n");
		this.header = lines[0];
		this.suggestedHeader = header;

		String body = "";

		if(lines.length > 1){
			for(int i = 1; i < lines.length; i++){
				body += lines[i];
			}
		}

		tags = new ArrayList<MessageTag>();

		tokeniseHeader();
	}

	public void tokeniseHeader(){
		Sentence headerSentence = new Sentence(header);
		headerTokens = new ArrayList<String>(headerSentence.words());
		headerPosTags = new ArrayList<String>(headerSentence.posTags());

		BracketsFixer.correctBrackets(headerTokens);
	}

	public String getHeader(){
		return header;
	}

	public String getBody(){
		return body;
	}

	public List<String> getHeaderTokens(){
		return headerTokens;
	}

	public List<String> getHeaderPosTags(){
		return headerPosTags;
	}

	public ArrayList<MessageTag> getTags(){
		return tags;
	}

	public void addTag(MessageTag tag){
		tags.add(tag);
	}

	public void printTags(){
		System.out.println(header);

		for(MessageTag tag : tags){
			System.out.println(tag.toString());
		}
	}

	public String getSuggestedHeader(){
		return suggestedHeader;
	}
```

```

    public String getSuggestedBody(){
        return suggestedBody;
    }

    public void setSuggestedHeader(String suggestedHeader){
        this.suggestedHeader = suggestedHeader;
    }

    public void setSuggestedBody(String suggestedBody){
        this.suggestedBody = suggestedBody;
    }

    public String toString(){
        return String.format("%s | SHA: %s", header, sha);
    }

    public String getSha(){
        return sha;
    }

    public void generateSuggestions(){
        String headerSuggestion = "";

        for(int i = 0; i < headerTokens.size(); i++){
            boolean addSpace = true;

            headerSuggestion += headerTokens.get(i);

            if(i + 1 < headerTokens.size()){
                if(TokenChecker.isPunctuationToken(headerPosTags.get(i + 1)) || TokenChecker.isClosingBracket(headerTokens.get(i + 1))){
                    addSpace = false;
                }
            }

            if(TokenChecker.isBracket(headerTokens.get(i))){
                addSpace = false;
            }

            if(i == headerTokens.size() - 1){
                addSpace = false;
            }

            if(addSpace){
                headerSuggestion += " ";
            }
        }

        suggestedHeader = headerSuggestion;
    }
}
```

MessageTag.java

```
import java.util.ArrayList;

public class MessageTag {
    private String tagMessage;
    private String suggestedChange;
    private boolean positive;

    public MessageTag(String tagMessage, boolean positive){
        this.tagMessage = tagMessage;
        this.positive = positive;
    }

    public MessageTag(String tagMessage, String suggestedChange, boolean positive){
        this.tagMessage = tagMessage;
        this.suggestedChange = suggestedChange;
        this.positive = positive;
    }

    public String getTagMessage(){
        return this.tagMessage;
    }

    public String getSuggestedChange(){
        return this.suggestedChange;
    }

    public boolean isPositive(){
        return this.positive;
    }

    public String toString(){
        String str = (positive ? "+ " : "- ");
        str += tagMessage;

        return str;
    }
}
```

MessageTagger.java

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

```java

public class MessageTagger {
    private static int messagesAnalysed = 0;

    public static void resetCounts(){
        messagesAnalysed = 0;
        HeaderLengthTagger.resetCount();
        HeaderVerbOrderTagger.resetCount();
        HeaderVerbTenseTagger.resetCount();
        HeaderGrammarTagger.resetCount();
        HeaderPunctuationTagger.resetCount();
        HeaderVaguenessTagger.resetCount();
    }

    public static void generateTags(CommitMessage message){
        messagesAnalysed++;
        HeaderLengthTagger.tagMessage(message);
        HeaderVerbOrderTagger.tagMessage(message);
        HeaderVerbTenseTagger.tagMessage(message);
        HeaderGrammarTagger.tagMessage(message);
        HeaderPunctuationTagger.tagMessage(message);
        HeaderVaguenessTagger.tagMessage(message);

        message.generateSuggestions();
    }

    public static void main(String[] args){
        new MainWindow();
    }

    public static int getMessagesAnalysed(){
        return messagesAnalysed;
    }
}
```

MessageGetter.java

```java
import com.mashape.unirest.http.*;
import org.json.JSONArray;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.regex.Pattern;
import java.io.File;
import java.io.BufferedReader;
import java.io.FileReader;

public class MessageGetter{
    private static String username = "mprzewlocki98";
    private static String OAuthToken = "3162fc99a81334029b8cfdcfaaba377d702a4f2b";
    public static String GithubAPIUrl = "https://api.github.com";

       public static String currentRepository = "";

       public static String[] parseUrl(String repositoryUrl){
           String newUrl = repositoryUrl.replace("https://github.com/", "");
           String[] parsedUrl = newUrl.split("/");
           return parsedUrl;
       }

       public static JSONObject getCommitMessageData(String repositoryUrl, String sha){
           String[] parsedUrl = parseUrl(repositoryUrl);
           String username = parsedUrl[0];
           String repo = parsedUrl[1];

           JSONObject object = new JSONObject();

           try{
               HttpResponse<JsonNode> firstResponse = Unirest.get(GithubAPIUrl + "/repos/" + username + "/" + repo + "/commits/" + sha).basicAuth(username, OAuthToken).asJson();

               object = firstResponse.getBody().getObject();
           }catch(Exception e){

           }

           return object;
       }

       public static ArrayList<CommitMessage> getCommitMessagesFromRepository(String repositoryUrl){
           currentRepository = repositoryUrl;

           ArrayList<CommitMessage> messages = new ArrayList<CommitMessage>();

           String[] parsedUrl = parseUrl(repositoryUrl);
           String username = parsedUrl[0];
           String repo = parsedUrl[1];

           try{
               HttpResponse<JsonNode> firstResponse = Unirest.get(GithubAPIUrl + "/repos/" + username + "/" + repo + "/commits?per_page=300").basicAuth(username, OAuthToken).asJson();

               JSONArray array = firstResponse.getBody().getArray();

               for(int i = 0; i < array.length(); i++){
                   String sha = array.getJSONObject(i).getString("sha");
                   String message = array.getJSONObject(i).getJSONObject("commit").getString("message");
                   messages.add(new CommitMessage(message, sha));
               }
           }catch(Exception e){

           }

           return messages;
       }

       public static ArrayList<CommitMessage> getCommitMessagesFromFile(String filePath){
           currentRepository = "";

           ArrayList<CommitMessage> messages = new ArrayList<CommitMessage>();

           try{
               File file = new File(filePath);
               BufferedReader br = new BufferedReader(new FileReader(file));
               String st = br.readLine();

               while(st != null){
                   CommitMessage message = new CommitMessage(st);
                   messages.add(message);
                   st = br.readLine();
               }
           }catch(Exception e){
               e.printStackTrace();
           }

           return messages;
       }
   }
```

BracketsFixer.java

```
import java.util.HashMap;
import java.util.ArrayList;

public class BracketsFixer {
    private static HashMap<String, String> tokensToBrackets = new HashMap<String, String>();
    private static boolean generatedMap = false;
```

```java

    public static void generateMap(){
        tokensToBrackets.put("-LRB-", "(");
        tokensToBrackets.put("-RRB-", ")");
        tokensToBrackets.put("-LCB-", "{");
        tokensToBrackets.put("-RCB-", "}");
        tokensToBrackets.put("-LSB-", "[");
        tokensToBrackets.put("-RSB-", "]");

        generatedMap = true;
    }

    public static void correctBrackets(ArrayList<String> tokens){
        if(!generatedMap){
            generateMap();
        }

        for(int i = 0; i < tokens.size(); i++){
            String token = tokens.get(i);

            if(tokensToBrackets.get(token) != null){
                tokens.set(i, tokensToBrackets.get(token));
            }
        }
    }
}
```

TokenChecker.java

```java
import java.util.HashMap;

public class TokenChecker {
    private static HashMap<String, Boolean> createVerbTokens() {
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
        map.put("VB", true);
        map.put("VBD", true);
        map.put("VBG", true);
        map.put("VBN", true);
        map.put("VBP", true);
        map.put("VBZ", true);

        return map;
    }

    private static HashMap<String, Boolean> createNonImperativeVerbTokens() {
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
        map.put("VBD", true);
        map.put("VBG", true);
        map.put("VBN", true);
        map.put("VBZ", true);

        return map;
    }

    private static HashMap<String, Boolean> createNounTokens() {
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
        map.put("NN", true);
        map.put("NNS", true);

        return map;
    }

    private static HashMap<String, Boolean> createProperNounTokens() {
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
        map.put("NNP", true);
        map.put("NNPS", true);

        return map;
    }

    private static HashMap<String, Boolean> createPunctuationTokens() {
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
        map.put("#", true);
        map.put("$", true);
        map.put(".", true);
        map.put(",", true);
        map.put(":", true);
        map.put("'", true);

        return map;
    }

    private static HashMap<String, Boolean> createBrackets() {
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
        map.put("(", true);
        map.put(")", true);
        map.put("{", true);
        map.put("}", true);
        map.put("[", true);
        map.put("]", true);

        return map;
    }

    private static HashMap<String, Boolean> createClosingBrackets() {
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
        map.put(")", true);
        map.put("}", true);
        map.put("]", true);

        return map;
    }

    private static HashMap<String, Boolean> verbTokens = createVerbTokens();
    private static HashMap<String, Boolean> nonImperativeVerbTokens = createNonImperativeVerbTokens();
    private static HashMap<String, Boolean> nounTokens = createNounTokens();
    private static HashMap<String, Boolean> properNounTokens = createProperNounTokens();
    private static HashMap<String, Boolean> punctuationTokens = createPunctuationTokens();
    private static HashMap<String, Boolean> brackets = createBrackets();
    private static HashMap<String, Boolean> closingBrackets = createClosingBrackets();

    public static boolean isVerbToken(String token){
        if(verbTokens.get(token) != null){
            return true;
        }else{
            return false;
        }
    }

    public static boolean isNonImperativeVerbToken(String token){
        if(nonImperativeVerbTokens.get(token) != null){
            return true;
        }else{
            return false;
        }
    }

    public static boolean isNounToken(String token){
        if(nounTokens.get(token) != null){
            return true;
```

```
		}

		if(properNounTokens.get(token) != null){
			return true;
		}

		return false;
	}

	public static boolean isProperNounToken(String token){
		if(properNounTokens.get(token) != null){
			return true;
		}else{
			return false;
		}
	}

	public static boolean isNonProperNounToken(String token){
		if(verbTokens.get(token) != null){
			return true;
		}else{
			return false;
		}
	}

	public static boolean isPunctuationToken(String token){
		if(punctuationTokens.get(token) != null){
			return true;
		}else{
			return false;
		}
	}

	public static boolean isBracket(String token){
		if(brackets.get(token) != null){
			return true;
		}else{
			return false;
		}
	}

	public static boolean isClosingBracket(String token){
		if(closingBrackets.get(token) != null){
			return true;
		}else{
			return false;
		}
```

```
        }
    }
```

VagueNounChecker.java

```
import java.util.HashMap;

public class VagueNounChecker {
	private static HashMap<String, Boolean> vagueNouns = new HashMap<String, Boolean>();
	private static boolean generatedList = false;

	public static void generateVagueNounList(){
		vagueNouns.put("bug", true);
		vagueNouns.put("bugs", true);
		vagueNouns.put("feature", true);
		vagueNouns.put("features", true);
		vagueNouns.put("it", true);
		vagueNouns.put("thing", true);
		vagueNouns.put("things", true);
		vagueNouns.put("stuff", true);

		generatedList = true;
	}

	public static boolean isVagueNoun(String noun){
		if(!generatedList){
			generateVagueNounList();
		}

		String n = noun.toLowerCase();

		return (vagueNouns.get(n) != null);
	}
}
```

Tagger.java

```
public interface Tagger {
	public static void tagMessage(CommitMessage message){}
	public static int getCount(){ return 0; }
	public static void resetCount(){}
}
```

HeaderGrammarTagger.java

```
import edu.stanford.nlp.simple.*;
import java.lang.Character;
import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;

public class HeaderGrammarTagger implements Tagger {
	private static int count = 0;
```

```

    private static boolean checkIsCapitalised(String word){
        return Character.isUpperCase(word.charAt(0));
    }

    public static String setCapitalised(String word, boolean capitalised){
        String w = word.toLowerCase();

        if(capitalised){
            char c = Character.toUpperCase(word.charAt(0));
            w = c + w.substring(1);
        }

        return w;
    }

    private static boolean checkNonProperNounGrammar(CommitMessage message){
        boolean incorrectGrammar = false;

        List<String> tokens = message.headerTokens;
        List<String> posTags = message.headerPosTags;

        for(int i = 0; i < tokens.size(); i++){
            if(!TokenChecker.isProperNounToken(posTags.get(i))){
                if(checkIsCapitalised(tokens.get(i))){
                    if(i > 0){
                        incorrectGrammar = true;
                        message.headerTokens.set(i, setCapitalised(message.headerTokens.get(i), false));
                    }
                }else{
                    if(i == 0){
                        incorrectGrammar = true;
                        message.headerTokens.set(i, setCapitalised(message.headerTokens.get(i), true));
                    }
                }
            }
        }

        if(incorrectGrammar){
            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);
            message.addTag(tag);
        }

        return incorrectGrammar;
    }

    private static boolean checkProperNounGrammar(CommitMessage message){
        boolean incorrectProperNounGrammar = false;

        List<String> tokens = message.headerTokens;
        List<String> posTags = message.headerPosTags;

        for(int i = 0; i < tokens.size(); i++){
            if(TokenChecker.isProperNounToken(posTags.get(i))){
                if(!checkIsCapitalised(tokens.get(i))){
                    incorrectProperNounGrammar = true;

                    message.headerTokens.set(i, setCapitalised(message.headerTokens.get(i), true));
                }
            }
        }

        if(incorrectProperNounGrammar){
            MessageTag tag = new MessageTag("One or more proper nouns in the message is incorrectly capitalised - proper nouns should always be capitalised", false);
            message.addTag(tag);
        }

        return incorrectProperNounGrammar;
    }

    public static void tagMessage(CommitMessage message){
        boolean incorrectNounGrammar = checkNonProperNounGrammar(message);
        boolean incorrectProperNounGrammar = checkProperNounGrammar(message);

        if(!incorrectNounGrammar && !incorrectProperNounGrammar){
            MessageTag tag = new MessageTag("The message has correct grammar", true);
            message.addTag(tag);
        }else{
            count++;
        }
    }

    public static int getCount(){
        return count;
```

```
	}

	public static void resetCount(){
		count = 0;
	}
}
```

HeaderLengthTagger.java

```
public class HeaderLengthTagger implements Tagger {
	private static int count = 0;

	public static void tagMessage(CommitMessage message){
		int length = message.getHeader().length();
		MessageTag tag;

		if(length < 50){
			tag = new MessageTag("The length of the header is fine", true);
		}else if(length >= 50 && length < 72){
			tag = new MessageTag("The length of the header may be too long", false);
			count++;
		}else{
			tag = new MessageTag("The length of the header is too long", false);
			count++;
		}

		message.addTag(tag);
	}

	public static int getCount(){
		return count;
	}

	public static void resetCount(){
		count = 0;
	}
}
```

HeaderPunctuationTagger.java

```
import edu.stanford.nlp.simple.*;
import java.util.List;

public class HeaderPunctuationTagger implements Tagger {
	private static int count = 0;
	private static String concatTokens(List<String> tokens, int upTo){
		String str = "";

		for(int i = 0; i < upTo; i++){
```

```
            str += tokens.get(i) + " ";
        }

        return str;
    }

    public static void tagMessage(CommitMessage message){
        List<String> tokens = message.headerTokens;
        List<String> posTags = message.headerPosTags;

        int index = posTags.size() - 1;

        String lastTag = posTags.get(index);

        if(TokenChecker.isPunctuationToken(lastTag)){
            String suggestion = concatTokens(tokens, tokens.size() - 1);
            MessageTag tag = new MessageTag("Header should not end with punctuation mark, as it is a title", suggestion, false);
            message.addTag(tag);

            message.headerTokens.remove(index);
            message.headerPosTags.remove(index);

            count++;
            return;
        }
    }

    public static int getCount(){
        return count;
    }

    public static void resetCount(){
        count = 0;
    }
}
```

HeaderVaguenessTagger.java

```
import edu.stanford.nlp.simple.*;
import org.json.JSONObject;
import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;

public class HeaderVaguenessTagger implements Tagger {
    private static int count = 0;

    private static boolean checkForVerb(CommitMessage message){
        for(String tag : message.headerPosTags){
            if(TokenChecker.isVerbToken(tag)){
                return true;
            }
        }

        return false;
    }

    private static boolean checkForNoun(CommitMessage message){
        for(String tag : message.headerPosTags){
            if(TokenChecker.isNounToken(tag)){
                return true;
            }
        }

        return false;
    }

    private static boolean checkForVagueNouns(CommitMessage message){
        int nouns = 0;
        int vagueNouns = 0;

        for(String tag : message.headerPosTags){
            if(TokenChecker.isNounToken(tag)){
                nouns++;
            }
        }

        for(String tag : message.headerTokens){
            if(VagueNounChecker.isVagueNoun(tag)){
                vagueNouns++;
            }
        }

        return (vagueNouns >= nouns && nouns > 0);
    }

    private static String getFileStatus(CommitMessage message){

    if(MessageGetter.currentRepository.equals("")){ return ""; }
        if(message.getSha().equals("")){ return ""; }

        JSONObject messageData = MessageGetter.getCommitMessageData(MessageGetter.currentRepository, message.getSha());
```

```
            String firstFileStatus = messageData.getJSONArray("files").getJSONObject(0).getString("status");

            return new Sentence(firstFileStatus).lemma(0);
        }

        private static String getFileName(CommitMessage message){

        if(MessageGetter.currentRepository.equals("")){ return ""; }
            if(message.getSha().equals("")){ return ""; }

            JSONObject messageData = MessageGetter.getCommitMessageData(MessageGetter.currentRepository, message.getSha());
            String firstFile = messageData.getJSONArray("files").getJSONObject(0).getString("filename");

            return firstFile;
        }

        public static void tagMessage(CommitMessage message){
            boolean hasVerb = checkForVerb(message);
            boolean hasNoun = checkForNoun(message);
            boolean hasOnlyVagueNouns = checkForVagueNouns(message);

            if(!hasVerb && !hasNoun){
                MessageTag tag = new MessageTag("Message is too vague: does not describe what change was made at all", "add a verb and a noun", false);
                message.addTag(tag);

                count++;
            }else if(!hasVerb){
                MessageTag tag = new MessageTag("Message is too vague: does not contain any meaningful verbs", "add a verb", false);
                message.addTag(tag);

                if(hasOnlyVagueNouns){
                    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);
                    message.addTag(tag2);
                }

                count++;
```

```java
        }else if(!hasNoun){
            MessageTag tag = new MessageTag("Message is too vague: does not contain any meaningful nouns", "add a noun", false);
            message.addTag(tag);

            count++;
        }else if(hasOnlyVagueNouns){
            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);
            message.addTag(tag);

            count++;
        }
        }

        public static int getCount(){
            return count;
        }

        public static void resetCount(){
            count = 0;
        }
    }
```

HeaderVerbOrderTagger.java

```java
import edu.stanford.nlp.simple.*;
import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;

public class HeaderVerbOrderTagger implements Tagger {
    private static int count = 0;

    public static void tagMessage(CommitMessage message){
        boolean hasIncorrectOrder = false;
        int firstVerbIndex = -1;

        ArrayList<String> tokens = message.headerTokens;
        ArrayList<String> posTags = message.headerPosTags;

        if(!TokenChecker.isVerbToken(posTags.get(0))){
            hasIncorrectOrder = true;
        }

        for(int i = 0; i < posTags.size(); i++){
            if(TokenChecker.isVerbToken(posTags.get(i))){
                firstVerbIndex = i;
                break;
            }
        }

        if(hasIncorrectOrder){
            MessageTag tag = new MessageTag("The message should start with a verb.", false);
            message.addTag(tag);

            if(firstVerbIndex > 0){
                String firstVerb = tokens.get(firstVerbIndex);
                String firstVerbPosTag = posTags.get(firstVerbIndex);

                tokens.remove(firstVerbIndex);
                posTags.remove(firstVerbIndex);

                tokens.add(0, firstVerb);
                posTags.add(0, firstVerbPosTag);
            }

            count++;
        }
    }

    public static int getCount(){
        return count;
    }

    public static void resetCount(){
        count = 0;
    }
}
```

HeaderVerbTenseTagger.java

```java
import edu.stanford.nlp.simple.*;
import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;

public class HeaderVerbTenseTagger implements Tagger {
    private static int count = 0;

    private static String getBaseForm(String word){
        String w = new Sentence(word).lemma(0);

        return w;
    }

    public static void tagMessage(CommitMessage message){
        boolean hasVerbs = false;
```

```
            boolean hasIncorrectTense = false;
            boolean chcekTense = true;

            List<String> tokens = message.headerTokens;
            List<String> posTags = message.headerPosTags;

            for(int i = 0; i < tokens.size(); i++){
                  String tag = posTags.get(i);

                 if(tag.equals(“RP”) || tag.equals(“IN”)){
                     checkTense = false;
                 }

                  if(TokenChecker.isVerbToken(tag) &&
checkTense){
                        hasVerbs = true;
                  }

                  if(TokenChecker.isNonImperativeVerbToken(tag)
&& checkTense){
                        hasIncorrectTense = true;
                        message.headerTokens.set(i,
getBaseForm(message.headerTokens.get(i)));
                  }
            }

            if(hasIncorrectTense){
                  MessageTag tag = new MessageTag("Verbs should
be in the present imperative form", false);
                  message.addTag(tag);

                  count++;
            }else if(hasVerbs){
                  MessageTag tag = new MessageTag("All verbs are
in the present imperative form.", true);
                  message.addTag(tag);
            }
      }

      public static int getCount(){
            return count;
      }

      public static void resetCount(){
            count = 0;
      }
}
```

MainWindow.java

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

```java
import java.awt.Dimension;
import java.util.ArrayList;

public class MainWindow {
	public MainWindow () {
		JFrame frame = new JFrame();
		frame.setSize(500, 300);
		frame.setMinimumSize(new Dimension(500, 300));
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		JPanel panel = new JPanel();
		BoxLayout layout = new BoxLayout(panel, BoxLayout.Y_AXIS);
		panel.setLayout(layout);

		JLabel welcome = new JLabel("Welcome to Git commit message analysis through natural language processing");
		JLabel inputMessage = new JLabel("Please input a commit message for analysis:");
		JLabel inputRepo = new JLabel("or type the web address of a repository:");
		JLabel inputFile = new JLabel("or type the address of a file:");

		JTextField singleMessageTextField = new JTextField();
		JButton singleMessageButton = new JButton("Analyse single message");

		singleMessageButton.addActionListener(new ActionListener(){
			@Override
			public void actionPerformed(ActionEvent e){
				CommitMessage message = new CommitMessage(singleMessageTextField.getText());
				MessageTagger.generateTags(message);
				new FeedbackWindow(message);
			}
		});

		JTextField repoTextField = new JTextField();
		JButton repoButton = new JButton("Analyse repository");

		repoButton.addActionListener(new ActionListener(){
			@Override
			public void actionPerformed(ActionEvent e){
				MessageTagger.resetCounts();

```

```
                ArrayList<CommitMessage> messages = MessageGetter.getCommitMessagesFromRepository(repoTextField.getText());

                messages.parallelStream().forEach(message -> MessageTagger.generateTags(message));
                new FeedbackList(messages);
            }
        });

        JTextField fileTextField = new JTextField();
        JButton fileButton = new JButton("Analyse file");

        fileButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                MessageTagger.resetCounts();

                ArrayList<CommitMessage> messages = MessageGetter.getCommitMessagesFromFile(fileTextField.getText());

                messages.parallelStream().forEach(message -> MessageTagger.generateTags(message));
                new FeedbackList(messages);
            }
        });

        panel.add(welcome);
        panel.add(inputMessage);
        panel.add(singleMessageTextField);
        panel.add(singleMessageButton);
        panel.add(inputRepo);
        panel.add(repoTextField);
        panel.add(repoButton);
        panel.add(inputFile);
        panel.add(fileTextField);
        panel.add(fileButton);

        frame.add(panel);
        frame.setVisible(true);
    }
}
```

FeedbackWindow.java

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class FeedbackWindow {
```

```java
    public static ArrayList<String> getTagsAsString(CommitMessage message){
        ArrayList<String> list = new ArrayList<String>();

        for(MessageTag tag : message.getTags()){
            list.add(tag.toString());
        }

        return list;
    }

    public FeedbackWindow(CommitMessage message) {
        JFrame frame = new JFrame("Commit message feedback");
        frame.setSize(500, 200);

        JPanel panel = new JPanel(new BorderLayout());
        frame.add(panel);

        JLabel label = new JLabel(message.getHeader());
        panel.add(label, BorderLayout.PAGE_START);

        DefaultListModel<String> listModel = new DefaultListModel<String>();
        ArrayList<String> tagStringList = getTagsAsString(message);

        for(String s : tagStringList){
            listModel.addElement(s);
        }

        JList<String> list = new JList<String>(listModel);
        panel.add(new JScrollPane(list), BorderLayout.CENTER);

        JLabel suggestedChange = new JLabel("Suggested new message: " + message.getSuggestedHeader());
        panel.add(suggestedChange, BorderLayout.PAGE_END);

        frame.setVisible(true);
    }
}
```

FeedbackList.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class FeedbackList {
       public FeedbackList(ArrayList<CommitMessage> messages) {

           JFrame frame = new JFrame("Repository feeback");
           frame.setSize(500, 800);

           JPanel panel = new JPanel(new BorderLayout());
           frame.add(panel);

           JLabel label = new JLabel("Double click on a message to see the feedback for that message");
           panel.add(label, BorderLayout.PAGE_START);

           DefaultListModel<CommitMessage> listModel = new DefaultListModel<CommitMessage>();

           for(CommitMessage message : messages){
               listModel.addElement(message);
           }

           JList<CommitMessage> list = new JList<CommitMessage>(listModel);

           list.addMouseListener(new MouseAdapter() {
               public void mouseClicked(MouseEvent evt){
                       if(evt.getClickCount() == 2){
                           int index = list.locationToIndex(evt.getPoint());
                           CommitMessage message = listModel.getElementAt(index);

                           new FeedbackWindow(message);
                       }
               }
           });

           panel.add(new JScrollPane(list), BorderLayout.CENTER);

           JButton analyticsButton = new JButton("Analytics");

           analyticsButton.addActionListener(new ActionListener(){
               @Override
               public void actionPerformed(ActionEvent e){
                      new AnalyticsWindow();
               }
           });

           panel.add(analyticsButton, BorderLayout.SOUTH);

           frame.setVisible(true);
```

```
	}
}
```

AnalyticsWindow.java

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class AnalyticsWindow {
	public AnalyticsWindow() {
		JFrame frame = new JFrame("Analytics");
		frame.setSize(600, 200);

		JPanel panel = new JPanel();
		BoxLayout layout = new BoxLayout(panel, BoxLayout.Y_AXIS);
		panel.setLayout(layout);

		JLabel messagesAnalysed = new JLabel("Messages analysed: " + MessageTagger.getMessagesAnalysed());

		JLabel headerData = new JLabel("- Message header data -");
		JLabel lengthTags = new JLabel("Messages with incorrect header length: " + HeaderLengthTagger.getCount());
		JLabel grammarTags = new JLabel("Messages with incorrect grammar: " + HeaderGrammarTagger.getCount());
		JLabel punctuationTags = new JLabel("Messages with incorrect punctuation: " + HeaderPunctuationTagger.getCount());
		JLabel verbTenseTags = new JLabel("Messages with incorrect verb tense: " + HeaderVerbTenseTagger.getCount());
		JLabel verbOrderTags = new JLabel("Messages with incorrect verb order: " + HeaderVerbOrderTagger.getCount());
		JLabel vagueTags = new JLabel("Messages that are considered vague: " + HeaderVaguenessTagger.getCount());

		panel.add(messagesAnalysed);
		panel.add(headerData);
		panel.add(lengthTags);
		panel.add(grammarTags);
		panel.add(punctuationTags);
		panel.add(verbTenseTags);
		panel.add(verbOrderTags);
		panel.add(vagueTags);

		frame.add(panel);

		frame.setVisible(true);
	}
}
```

Part 2 – unit tests

GrammarTaggerTests.java

```java
import org.junit.*;
import static org.junit.Assert.assertTrue;

public class GrammarTaggerTests {
	@Test
	public void TestGrammarCorrect(){
		CommitMessage message = new CommitMessage("This has correct grammar");
		String tagMessage =  "The message has correct grammar";

		HeaderGrammarTagger.tagMessage(message);

		boolean hasTag = false;

		for(MessageTag tag : message.getTags()){
			if(tag.getTagMessage().equals(tagMessage)){
				hasTag = true;
			}
		}

		assertTrue(hasTag);
	}

	@Test
	public void TestGrammarIncorrectNoun(){
		CommitMessage message = new CommitMessage("This has incorrect Grammar");
		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";

		HeaderGrammarTagger.tagMessage(message);

		boolean hasTag = false;

		for(MessageTag tag : message.getTags()){
			if(tag.getTagMessage().equals(tagMessage)){
				hasTag = true;
			}
		}

		assertTrue(hasTag);
	}

	@Test
	public void TestGrammarIncorrectProperNoun(){
        CommitMessage message = new CommitMessage("Fix london");
        String tagMessage =  "One or more proper nouns in the message is incorrectly capitalised - proper nouns should always be capitalised";

        HeaderGrammarTagger.tagMessage(message);

        boolean hasTag = false;

        for(MessageTag tag : message.getTags()){
            if(tag.getTagMessage().equals(tagMessage)){
                hasTag = true;
            }
        }

        assertTrue(hasTag);
    }
}
```

LengthTaggerTests.java

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

public class LengthTaggerTests {
    @Test
    public void TestHeaderLengthTaggerOK(){
        CommitMessage OKMessage = new CommitMessage("This message is below 50 characters");
        String lengthMessage =  "The length of the header is fine";

        HeaderLengthTagger.tagMessage(OKMessage);

        boolean hasTag = false;

        for(MessageTag tag : OKMessage.getTags()){
            if(tag.getTagMessage().equals(lengthMessage)){
                hasTag = true;
            }
        }

        assertTrue(hasTag);
    }

    @Test
    public void TestHeaderLengthTaggerDubious(){
        CommitMessage dubiousMessage = new CommitMessage("This message is over 50 characters but not over 72");
```

```
            String lengthMessage = "The length of the header may be too long";

            HeaderLengthTagger.tagMessage(dubiousMessage);

            boolean hasTag = false;

            for(MessageTag tag : dubiousMessage.getTags()){
                  if(tag.getTagMessage().equals(lengthMessage)){
                        hasTag = true;
                  }
            }

            assertTrue(hasTag);
      }

      @Test

      public void TestHeaderLengthTaggerTooLong(){
            CommitMessage tooLongMessage = new CommitMessage("This message is definitely far too long and should be tagged accordingly.");
            String lengthMessage = "The length of the header is too long";

            HeaderLengthTagger.tagMessage(tooLongMessage);

            boolean hasTag = false;

            for(MessageTag tag : tooLongMessage.getTags()){
                  if(tag.getTagMessage().equals(lengthMessage)){
                        hasTag = true;
                  }
            }

            assertTrue(hasTag);
      }
}
```

PunctuationTaggerTests.java

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

public class PunctuationTaggerTests {
      @Test
      public void TestPunctuationTaggerFindssFinalPunctuation(){
            CommitMessage message = new CommitMessage("There should be no punctuation.");
            String tagMessage =  "Header should not end with punctuation mark, as it is a title";
```

```

        HeaderPunctuationTagger.tagMessage(message);

        boolean hasTag = false;

        for(MessageTag tag : message.getTags()){
            if(tag.getTagMessage().equals(tagMessage)){
                hasTag = true;
            }
        }

        assertTrue(hasTag);
    }
}
```

VaguenessTaggerTests.java

```
import org.junit.*;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertFalse;

public class VaguenessTaggerTests {
    @Test
    public void TestNonVagueMessage(){
        CommitMessage message = new CommitMessage("Add Polish translation");
        String tagMessage1 =  "Message is too vague: does not describe what change was made at all";
        String tagMessage2 = "Message is too vague: does not contain any meaningful verbs";
        String tagMessage3 = "Message is too vague: all nouns in the message are considered vague. Consider using more descriptive nouns";
        String tagMessage4 = "Message is too vague: does not contain any meaningful nouns";

        HeaderVaguenessTagger.tagMessage(message);

        boolean hasTag = false;

        for(MessageTag tag : message.getTags()){
            if(tag.getTagMessage().equals(tagMessage1)){
                hasTag = true;
            }

            if(tag.getTagMessage().equals(tagMessage2)){
                hasTag = true;
            }

            if(tag.getTagMessage().equals(tagMessage3)){
                hasTag = true;
            }
```

```

                if(tag.getTagMessage().equals(tagMessage4)){
                    hasTag = true;
                }
            }

            assertFalse(hasTag);
        }

        @Test
        public void TestOverlyVagueMessage(){
            CommitMessage message = new CommitMessage("0");
            String tagMessage =  "Message is too vague: does not describe what change was made at all";

            HeaderVaguenessTagger.tagMessage(message);

            boolean hasTag = false;

            for(MessageTag tag : message.getTags()){
                if(tag.getTagMessage().equals(tagMessage)){
                    hasTag = true;
                }
            }

            assertTrue(hasTag);
        }

        @Test
        public void TestVagueNoVerbMessage(){
            CommitMessage message = new CommitMessage("Cosmetic improvements");
            String tagMessage =  "Message is too vague: does not contain any meaningful verbs";

            HeaderVaguenessTagger.tagMessage(message);

            boolean hasTag = false;

            for(MessageTag tag : message.getTags()){
                if(tag.getTagMessage().equals(tagMessage)){
                    hasTag = true;
                }
            }

            assertTrue(hasTag);
        }

        @Test
        public void TestVagueNoNounMessage(){
            CommitMessage message = new CommitMessage("Added");
            String tagMessage =  "Message is too vague: does not contain any meaningful nouns";

            HeaderVaguenessTagger.tagMessage(message);

            boolean hasTag = false;

            for(MessageTag tag : message.getTags()){
                if(tag.getTagMessage().equals(tagMessage)){
                    hasTag = true;
                }
            }

            assertTrue(hasTag);
        }

        @Test
        public void TestVagueNounsMessage(){
            CommitMessage message = new CommitMessage("Fix bug");
            String tagMessage =  "Message is too vague: all nouns in the message are considered vague. Consider using more descriptive nouns";

            HeaderVaguenessTagger.tagMessage(message);

                boolean hasTag = false;

                for(MessageTag tag : message.getTags()){
                    if(tag.getTagMessage().equals(tagMessage)){
                        hasTag = true;
                    }
                }

                assertTrue(hasTag);
            }
        }
```

VerbOrderTaggerTests.java

```
import org.junit.*;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertEquals;

public class VerbOrderTaggerTests {
    @Test
    public void TestFindsCorrectOrder(){
        CommitMessage message = new CommitMessage("remove bug");
```

```
		String tagMessage = "The message should start with a verb.";

		HeaderVerbOrderTagger.tagMessage(message);

		boolean hasTag = false;

		for(MessageTag tag : message.getTags()){
			if(tag.getTagMessage().equals(tagMessage)){
				hasTag = true;
			}
		}

		assertFalse(hasTag);
	}

	@Test
	public void TestFindsIncorrectOrder(){
		CommitMessage message = new CommitMessage("bug removed");
		String tagMessage = "The message should start with a verb.";

		HeaderVerbOrderTagger.tagMessage(message);

		boolean hasTag = false;

		for(MessageTag tag : message.getTags()){
			if(tag.getTagMessage().equals(tagMessage)){
				hasTag = true;
			}
		}

		assertTrue(hasTag);
	}

	@Test
	public void TestChangesIncorrectOrder(){
		CommitMessage message = new CommitMessage("bug removed");
		HeaderVerbOrderTagger.tagMessage(message);
		message.generateSuggestions();

		assertEquals(message.getSuggestedHeader(), "removed bug");
	}
}
```

VerbTenseTaggerTests.java

```
import org.junit.*;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertEquals;

public class VerbTenseTaggerTests {
    @Test
    public void TestFindsCorrectTense(){
        CommitMessage message = new CommitMessage("Fix bug");
        String tagMessage = "All verbs are in the present imperative form.";

        HeaderVerbTenseTagger.tagMessage(message);

        boolean hasTag = false;

        for(MessageTag tag : message.getTags()){
            if(tag.getTagMessage().equals(tagMessage)){
                hasTag = true;
            }
        }

        assertTrue(hasTag);
    }

    @Test
    public void TestFindsIncorrectTense(){
        CommitMessage message = new CommitMessage("Fixed bug");
        String tagMessage = "Verbs should be in the present imperative form";

        HeaderVerbTenseTagger.tagMessage(message);

        boolean hasTag = false;

        for(MessageTag tag : message.getTags()){
            if(tag.getTagMessage().equals(tagMessage)){
                hasTag = true;
            }
        }

        assertTrue(hasTag);
    }

    @Test
    public void TestChangesIncorrectTense(){
        CommitMessage message = new CommitMessage("Fixed bug");
        HeaderVerbTenseTagger.tagMessage(message);
        message.generateSuggestions();

```

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

HolisticTests.java

```
import org.junit.*;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertEquals;

public class HolisticTests {
	@Test
	public void TestMessage1(){
		CommitMessage message = new CommitMessage("fix bug.");
		MessageTagger.generateTags(message);
		message.generateSuggestions();
		assertEquals(message.getSuggestedHeader(), "Fix bug");
	}

	@Test
	public void TestMessage2(){
		CommitMessage message = new CommitMessage("translation added");
		MessageTagger.generateTags(message);
		message.generateSuggestions();
		assertEquals(message.getSuggestedHeader(), "Add translation");
	}

	@Test
	public void TestMessage3(){
		CommitMessage message = new CommitMessage("added music, added game, removed feature");
		MessageTagger.generateTags(message);
		message.generateSuggestions();
		assertEquals(message.getSuggestedHeader(), "Add music, add game, remove feature");
	}
}
```

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 fo 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/XSStrike - 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/Eugeny/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/eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee/eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee - Accessed 10/12/2018

Create e.e

marking fn e as #[main]

remove new line

change `sys.exit(0)` to `sys.exit('e')`

Optimize Go implementation

Add e.png

Update e.java

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

update

eeeeeeeeeeeeeeeeeeeeeeeeee parseeeeeeeeeeeeeeeeeeer

e.bmp

https://github.com/google/open-location-code - Accessed 10/12/2018

icons

Update Garmin build files and docs (#237)

Add max code length for JS implementation. (#242)

Update version for Swift (#244)

Add go benchmarks (#240)

Add C/C++ library benchmark (#231)

Add library version information for C implementation (#230)

Add "Positioning" permission to the app (#233)

Fix the "you can just tell them 8FXR" example. (#232)

Gonzus/cpp example output (#229)

```
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.

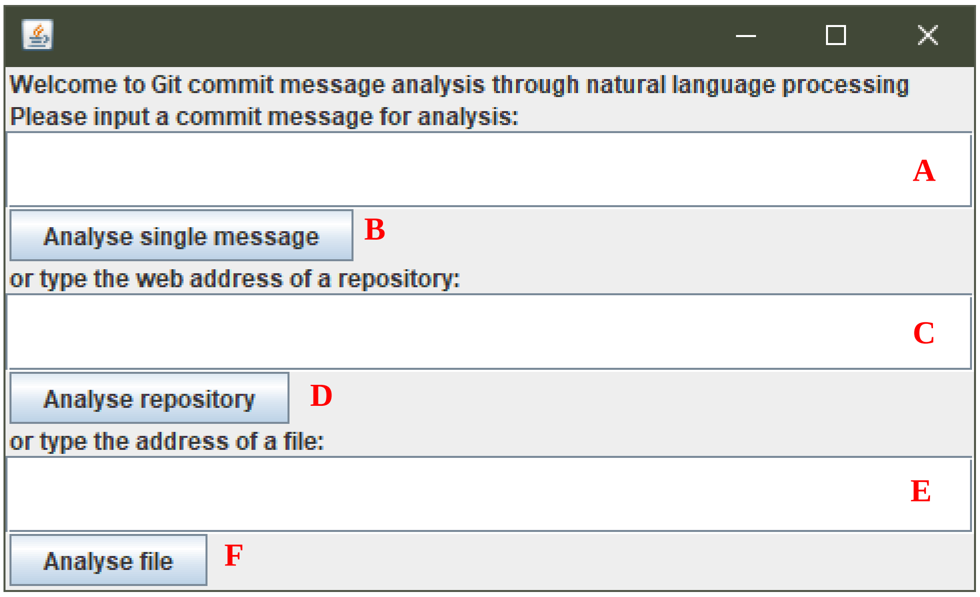
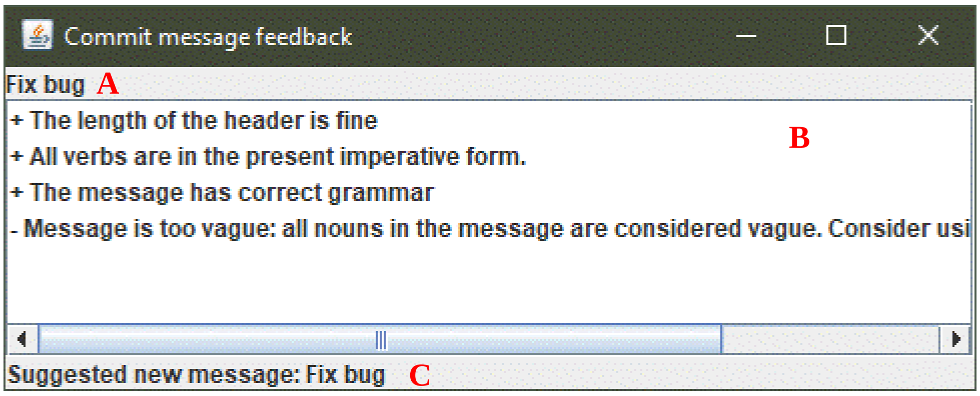

Welcome to Git commit message analysis through natural language processing
Please input a commit message for analysis:
or type the web address of a repository:
or type the address of a file:

Figure C.1 – initial application interface

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

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.



Commit message feedback

Fix bug A

```
+ The length of the header is fine
+ All verbs are in the present imperative form.
+ The message has correct grammar
- Message is too vague: all nouns in the message are considered vague. Consider usi
```

B

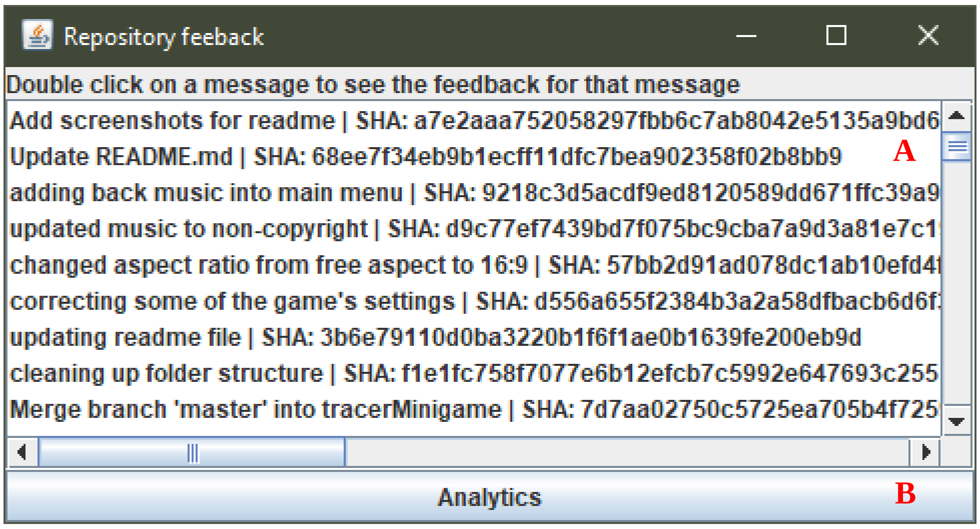
Suggested new message: Fix bug C

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.



Repository feeback

Double click on a message to see the feedback for that message

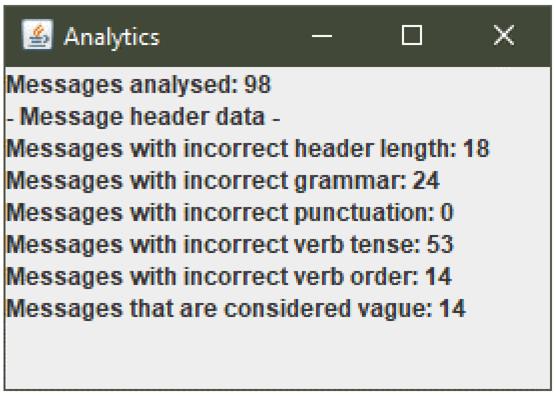
```
Add screenshots for readme | SHA: a7e2aaa752058297fbb6c7ab8042e5135a9bd6
Update README.md | SHA: 68ee7f34eb9b1ecff11dfc7bea902358f02b8bb9
adding back music into main menu | SHA: 9218c3d5acdf9ed8120589dd671ffc39a9
updated music to non-copyright | SHA: d9c77ef7439bd7f075bc9cba7a9d3a81e7c1
changed aspect ratio from free aspect to 16:9 | SHA: 57bb2d91ad078dc1ab10efd4
correcting some of the game's settings | SHA: d556a655f2384b3a2a58dfbacb6d6f
updating readme file | SHA: 3b6e79110d0ba3220b1f6f1ae0b1639fe200eb9d
cleaning up folder structure | SHA: f1e1fc758f7077e6b12efcb7c5992e647693c255
Merge branch 'master' into tracerMinigame | SHA: 7d7aa02750c5725ea705b4f725
```

A

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.



```
Messages analysed: 98
- Message header data -
Messages with incorrect header length: 18
Messages with incorrect grammar: 24
Messages with incorrect punctuation: 0
Messages with incorrect verb tense: 53
Messages with incorrect verb order: 14
Messages that are considered vague: 14
```

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.