Extracting Words in Tweets that Exemplify the Sentiment

Xiaoyan Su, Yuqi Shi, Jinwei Bi

Project Mentor TA: Sihao Chen
Github Link: https://github.com/Suxiaoyan98/tweet_sentiment_extraction

1) Abstract

For this project, We propose to look at the labeled sentiment for a given tweet and extract which words or phrases best reflect the provided sentiment. Not only sentiment extraction is important but figuring out which words actually lead to the sentiment description is also crucial. Our primary contribution is to convert the text data into vectors using pre-trained word embedding. Secondly, we would like to design LSTM and Bi-directional LSTM models to extract supported words. Then we also trained pre-trained BERT and RoBERTa to make comparisons. To evaluate the performance of our models, we will use a word-level Jaccard score, which gives the similarity between two sets of phrases or texts. The result shows that Bi-LSTM is better than LSTM and RoBERTa has the highest accuracy.

2) Introduction

Twitter is a microblogging service that allows users to share their life, express their opinion and post their status. Millions of users are posting millions of tweets everyday. Tweets are circulating every second. It will be interesting to determine the sentiment behind these tweets, which might be critical for a person, for a company and even for broader social and political impacts. Decisions and reactions are made in seconds after the tweets are sent out. Automatic tools can help decision-makers to ensure efficient solutions to the problems raised.

Sentiment analysis is a rather popular topic in Natural Language Processing. It is a process to categorize text into three types of sentiments (neutral, positive and negative).This technique is utilized in various scenarios such as social media monitoring, customer services and even political campaigns to understand peoples' needs and reactions. Many works have been done to predict sentiments. However, our work uses sentiment as an input and builds models to identify which words really determine the tone of the tweets. Our focus is to predict words or phrases that matter, instead of simply focusing on the sentiment of the tweets.

For our project, we propose to look at the labeled sentiment for a given tweet and extract which words or phrases that best reflect the provided sentiment. The input of our problem is a dataset of over 27,000 tweets, with its text id and its sentiment label (neutral, negative and positive). The output of our system is a word or a phrase that is part of the tweets that encapsulate the labeled sentiment.

After splitting our dataset into training and testing data, we will train our models and fine tune them. Our results will come from LSTM, Bi-directional LSTM, pre-trained BERT and RoBERTa models. To evaluate the performance of our models, we will use word-level Jaccard score, which gives the similarity between two sets of phrases or texts.

3) Background

For machine learning or deep learning, data wrangling is very essential to improve the effectiveness and avoid any unnecessary errors. Since the dataset we get are tweets from the web, which vary from person to person, we need to clean the data. We referred to the two articles below to help us preprocess our dataset.
Also, our final decision was to train LSTM, Bi-directional LSTM, BERT, and RoBERTa models for sentiment extraction and then evaluate their performances. Therefore, the prior work that we referred to was mainly based on these models and is shown as follows:

1. Nikhil Raj , "Emotion analysis of sentences", June 15, 2021. Starters Guide to Sentiment Analysis using Natural Language Processing
   https://www.analyticsvidhya.com/blog/2021/06/nlp-sentiment-analysis/.
   This blog is a starter guide on how to do exploratory data analysis and data preprocessing.
2. Aston Zhang, et al. "Dive into deep learning", 2021. Modern Recurrent Neural Networks:
   https://d2l.ai/chapter_recurrent-modern/lstm.html
   It introduces several NLP-related models like LSTM, RNN, etc.
3. Alammar, J. (2018, December 3). The illustrated bert, elmo, and co. (how nlp cracked transfer learning). The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) — Jay Alammar — Visualizing machine learning one concept at a time.
   https://jalammar.github.io/illustrated-bert/
   This blog illustrated BERT architecture.

Note: the prior works are all focusing on how to predict the sentiment of the text. However, our project's goal is to extract words or phrases from the text that support a given sentiment label, which is totally opposite from the previous. So our model/algorithm will be different.

4) Summary of Our Contributions

1. **Contribution(s) in Code:**
   a. Code to perform data preprocessing.
   b. Code to implement LSTM, BERT, and RoBERTa models for sentiment extraction.
2. **Contribution(s) in Application:** N/A
3. **Contribution(s) in Data:**
   a. Dropped missing values, removed meaningless punctuations, and correct misspellings
   b. Tokenize tweets using NLTK and vectorize tweets
   c. Generate n-grams to use to train models

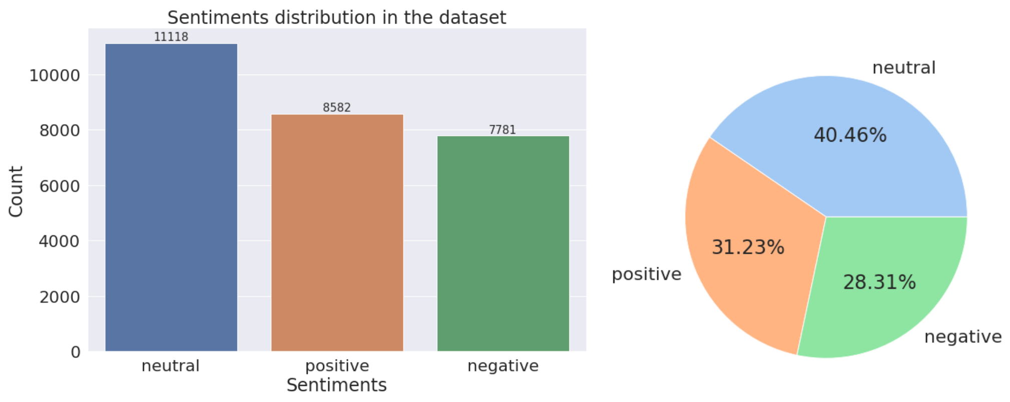
4. **Contribution(s) in Algorithm:** N/A
5. **Contribution(s) in Analysis:**
    a. We performed data visualization (histograms and word clouds) to give us an idea of the distribution of our dataset
    b. Before applying training data to models, we analyzed the pros and cons of different NLP models (LSTM, BERT, and RoBERTa)
    c. After training our models, we analyzed whether each model is good at generating the phrases in a text that express the sentiment

5) Detailed Description of Contributions

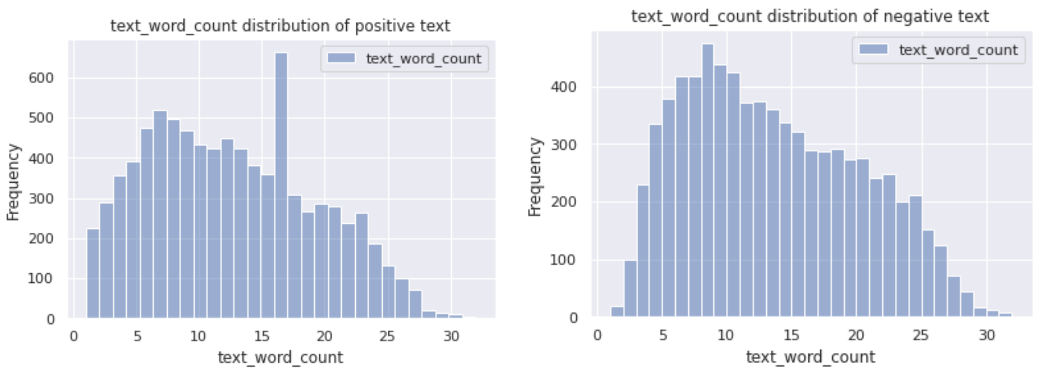
5.1 Methods

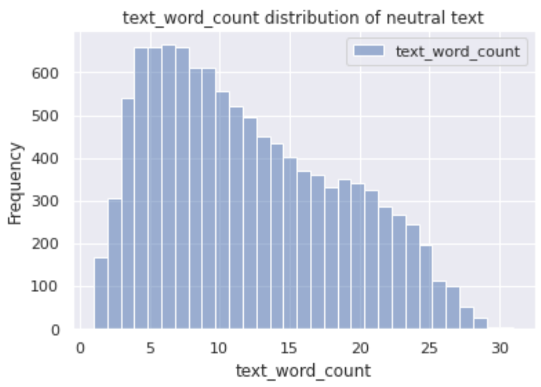
Contribution(s) in Data:

- Data analysis: We mainly used histograms to show the number distribution of texts. From the plot, we found that both training data and test data contain similar distributions. We used bar plots to analyze the average length of each sentiment text. We analyzed the length of the text with positive and negative sentiments and used word clouds to show the text's frequency. Some examples are shown as below:



Dataset distribution



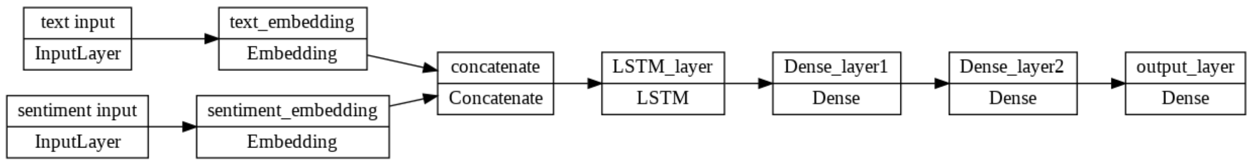


The frequency of text word count for each sentiment text

- Data preprocessing: The raw data which we have contains a lot of unnecessary things which can be called noise and it only hinders our algorithm's capability, so we will clean the data a little of this noise. We drop the missing value and nan from the training set. Removing URL links and useless information. Correcting spelling mistakes which include lowercase letters, eliminates words of length 1. After this, we get the clean reviews which can be used downstream for training.
- Tokenizing the words and splitting the data: We split our data into training, validation and test sets using the train_test_split function of the model_selection module from the sklearn library. The division is 80% of training data, 5% validation data, and 15% test data. Before implementing different neural network models, we need to convert the text data into vectors. We tokenize the 'text' and 'sentiment' data separately using the Tokenizer method in Tensorflow and padding all items into their max length(max length for text is 32/ max length for sentiment is 1). We extract the words_to_index of each tokenizer which is the dictionary mapping words to their respective index.

Contribution(s) in Code:

- Code to generate word embedding using GloVe: We defined the embedding layer using the built-in Keras Embedding layer. The embedding layer maps the words to their embedding vectors from the embedding matrix. If the words are not found in embedding, their index will be all-zeros. We keep this layer as not trainable, we are using GloVe Embeddings, and we don't want the neural network to learn the embedding matrix itself.
- Code to implement the LSTM and Bi-LSTM model for sentiment extraction: Our models are clearly defined in the colab. First, we use one embedding layer to map the text input data to their GloVe vectors, and another embedding layer to map the sentiment input data. Then we concatenate the two embeddings (dimension 32x300 and 1x300) together as the input vector to the LSTM layers followed by three Dense layers(relu, relu, sigmoid activation). Since adding drop out for LSTM cells, there is a chance of forgetting something that should not be forgotten. Consequently, we use to drop out in dense layers after the LSTM layers. The output is a vector of the length of input which is 33. It indicates which word will be selected from the 'text' that can represent the according to sentiment. The following is the general structure of our LSTM model:



The Bi-LSTM model is quite similar, we simply design the LSTM layer in bi-direction. The trick that we use to avoid overfitting is to do weight Regularization using L2-norm regularization. We set kernel_regularizer to l2 (lambda = 0.01) in Dense layers to add a penalty for weight size to the loss function.

- Code to implement the BERT and RoBERTa model for sentiment extraction: Our models are clearly defined in the colab. In order to apply the pre-trained BERT, we must use the provided tokenizer since the model has a specific, fixed vocabulary. In addition, we added special tokens to the start and end of each sentence, pad & truncate all sentences to the same length. We then created an iterator for our dataset using the torch DataLoader class. BERT-base consists of 12 transformer layers, each transformer layer takes in a list of token embeddings, and produces the same number of embeddings with the same hidden size (or dimensions) on the output. The output of the final transformer layer of the `[CLS]` token is used as the features of the sequence to feed a classifier.

Contribution(s) in Analysis:

- Different performance between LSTM and Bi-LSTM: With the regular LSTM, we can make input flow in one direction either backward or forward. However, in bi-directional we can make the input flow in both directions to preserve the future and the past information. BiLSTM adds one more LSTM layer, which reverses the direction of information flow. Every component of an input sequence has information from both the past and present. In our case, when we try to select the word according to the given sentiment, we need to understand the word in the context of the full sentence. For this reason, BiLSTM can produce a more meaningful output and our experiment result also prove this.
- Difference between BERT and RoBERTa and why RoBERTa has better outcome: As RoBERTa is developed based on BERT, we expected RoBERTa to produce a better output. BERT uses [CLS] and [SEP] as starting token and separator token respectively, on the other hand, RoBERTa uses <s> and </s> to covert sentences. RoBERTa also has around 20k more subwords and has larger training data.

5.2 Experiments and Results

Key Questions and Hypothesis

1. To extract words from the text that support the given sentiment. We directly visualize the result by converting the output vector from the neural network into dataframe.

2. To implement and analyze that Bi-LSTM model is better than LSTM when learning long-term dependencies.
3. To do a comparison of our rule-based method with the ones made by a supervised ML model such as BERT and RoBERTa.

Design Decision

Our model is trying to label the words in a given phrase with 0s and 1s. Instead of generating new words or phrases, we select words that represent the sentiment of the whole phrase. For example, someone posted "I love apple!" on Twitter, our implementation will first tokenize this tweet and model generate a sequence of 0s and 1s. If the model generated [1, 1, 0], that means "I love" is selected, indicating that it represents the positive tone of this tweet.

ML/DL Approaches

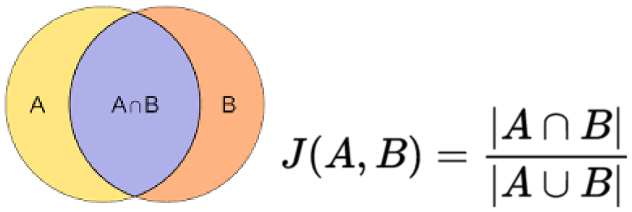
The baseline approach that we compared is the LSTM model in our own project for it is the most basic deep learning model used for sentiment analysis that deals with long strings. The other baseline we compare to for pre-trained models is the Pre-training BERT, which is the model trained on unlabeled data over different pre-training tasks.

Dataset

The dataset we evaluate is the test dataset we split from the raw data. We use the 'text' and 'sentiment' information to extract support phrases for sentimental labels. We evaluate our performance by calculating the Jaccard score on the whole test result as well as the positive sentiment results and negative sentiment results. We expected that the score for RoBERTa is the highest, then BERT and Bi-LSTM, and finally LSTM.

Performance Metrics

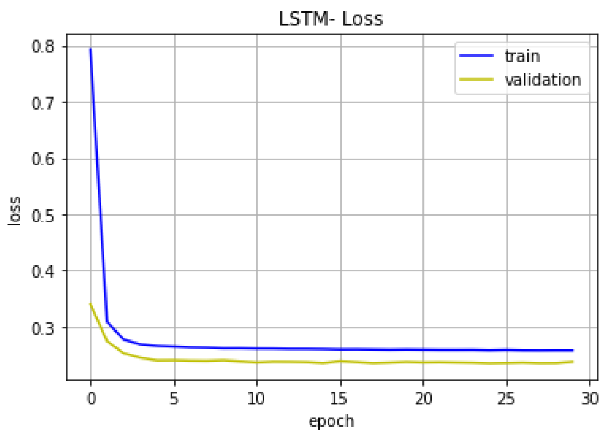
The metric in this problem is the word-level **Jaccard score**. Jaccard Score or Jaccard Similarity is defined as the size of the intersection divided by the size of the union of two sets. The formula is $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ where A, B are two label sets. For our model's results, we compare two lists that consist of 0s and 1s.

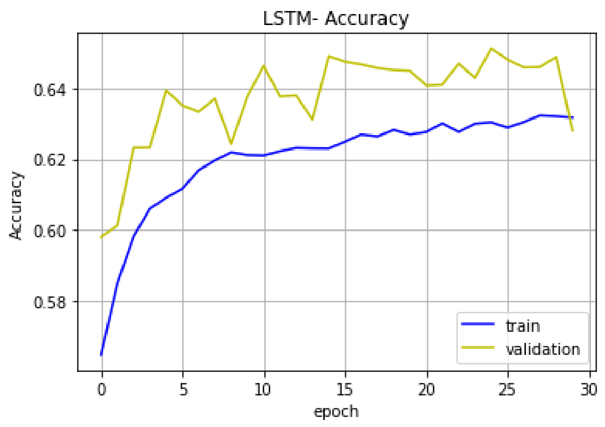


Jaccard score metrics

Results and Discussion

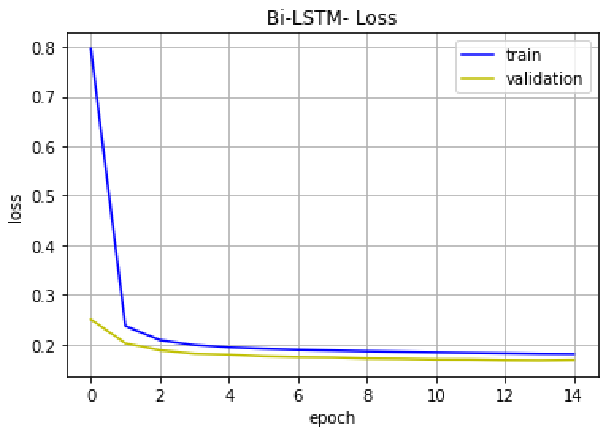
- For **LSTM** architecture, we have the results of loss/ accuracy on the training/ validation dataset as follows. Our prediction accuracy on the test dataset is about **0.55**

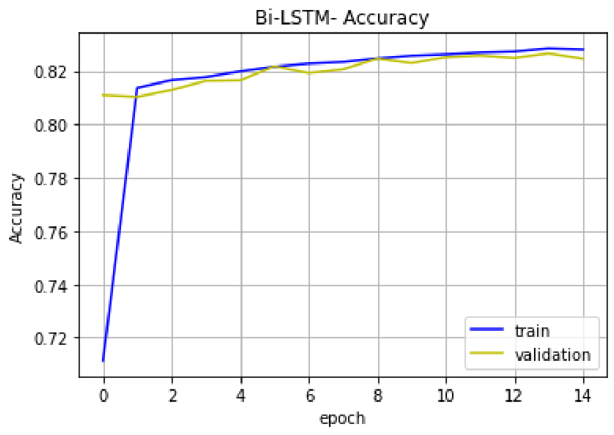




```
LSTM Jaccard score: 0.5483181312586016
LSTM jaccard score(positive sentiment): 0.336848343020183
LSTM jaccard score(negative sentiment) 0.3490257081741664
LSTM jaccard score(neutral sentiment) 0.8368355259562877
```

- For **Bi-LSTM** architecture, we have the results of loss/ accuracy on the training/ validation dataset as follows. Our prediction accuracy on the test dataset is about **0.63** which is higher than LSTM. It's obvious that BiLSTM can produce a more meaningful output.





```
Bi-LSTM Jaccard score: 0.6291808210853913
Bi-LSTM jaccard score(positive sentiment): 0.3727327888922423
Bi-LSTM jaccard score(negative sentiment) 0.37589760555632656
Bi-LSTM jaccard score(neutral sentiment) 0.9868850413324046
```

- For BERT model, our results of loss and Jaccard scores are shown below. We trained three epochs and Jaccard score increments with each epoch. Our prediction on the test dataset is around 0.6. Since it is the baseline, this is the result we expected to get. The limitation of our result is that we only trained the model with ¼ of the dataset. It takes around 1 and a half hours to go through an epoch.

BERT Loss: 0.18921947479248047
BERT Jaccard Score: 0.48732834200797004
BERT Loss: 0.17222631338870886
BERT Jaccard Score: 0.5806376112689436
BERT Loss: 0.16753120133371063
BERT Jaccard Score: 0.6178570029779169

- For RoBERTa model, our results of loss and Jaccard score is shown below. We trained three epochs and Jaccard score increments with each epoch. This TFRoBERTA model provides the best Jaccard score of 0.88 which is the BEST score among all models.
  RoBERTa Loss: 0.3483509583906694
  RoBERTa Jaccard Score: 0.7218303673943
  RoBERTa Loss: 0.22490169062758936
  RoBERTa Jaccard Score: 0.8787439355596516
  RoBERTa Loss: 0.21435484741673325
  RoBERTa Jaccard Score: 0.880919666948429

6) Conclusions

Outcomes

The main outcome of our project is that we achieved a relatively high score to predict the phrase that most represents the sentiment of a text. While doing this project, we are familiar with the specific process of doing an NLP project. We did exploratory data analysis and data preprocessing before training the dataset with some deep learning models like LSTM, BiLSTM, BERT and RoBERTa. We also compared them and analyzed their performance in terms of loss and Jaccard score. In conclusion, we found that RoBERTa model works the best.
Our result might be helpful to identify the critical content of a tweet and find the words or phrases that impact a sentence's sentiment the most. Our models could help to evaluate other models that predict word sentiments. For example, if a model irons out the sentiment of "wonderful" is positive, and we have predicted that "wonderful" in a sentence made the sentence positive. Our results, in this case, support their model. In addition, we think our data analysis and preprocessing part is relatively complete and comprehensive, which can be used for reference by other sentiment extraction NLP projects.

Idea Development

We don't have much experience in building those LSTM and BiLSTM models. In the feedback of the project mid-way report, TA also pointed out that LSTM requires more engineering effort than BERT and RoBERTa. So, we trained our dataset using the built-in LSTM and BiLSTM model in TensorFlow and fine-tuned the model to get a better Jaccard score. However, compared to BERT and RoBERTa models, the performance of LSTM models is not good enough.

Future Improvement
During the training process, we found that it was too time consuming to train the whole dataset with the BERT/RoBERTa models. To make sure that we can finish this project, we just trained ¼ of the dataset, which could badly affect our validation and test accuracy. For the future, I would recommend feeding in more dataset and more computing power to train the model. Moreover, we could use the Bertviz recommended by Sihao to visualize which tokens is more important for our label prediction.

Ethical Considerations, and Broader Social and Environmental Impact
Our datasets are from open resources, so they will be legal to use and apply to different models. As we mentioned above, the main takeaway of our project is to extract the phrase or words that best represent a given tweet according to its sentiment. Our models could help to evaluate other models that predict word sentiments. However, we only have about 27k data in our dataset, if our model is to build into applications or apply to other datasets, our model might have limitations due to the limited number of training data.

Other Prior Work / References (apart from Sec 3) that are cited in the text:

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL. https://arxiv.org/abs/1810.04805.

Tran, C. (n.d.). Fine-tuning bert for sentiment analysis. Chris Tran. Retrieved September 15, 2021, from https://chriskhanhtran.github.io/_posts/2019-12-25-bert-for-sentiment-analysis/.

Javaid Nabi, "Machine Learning — Word Embedding & Sentiment Classification using Keras", Oct 4, 2018, How to use deep learning architecture to create an improved Sentiment Classifier: https://towardsdatascience.com/machine-learning-word-embedding-sentiment-classification-using-keras-b83c28087456.

Broader Dissemination Information:

Your report title and the list of team members will be published on the class website. Would you also like your pdf report to be published?
NO

If your answer to the above question is yes, are there any other links to github / youtube / blog post / project website that you would like to publish alongside the report? If so, list them here.

Work Report:

PERSON (S)	TASK (S)	Wk1				Wk2				Wk3				Wk4				Wk5			
		FEB & MAR																	MAR		
		S	M	W	R	S	M	W	R	S	M	W	R	S	M	W	R	S	M	W	R
		20	21	23	24	27	28	02	03	06	07	09	10	13	14	16	17	20	21	23	24
Xiaoyan Su	Exploratory Data Analysis				🟩																
Jinwei Bi	Data preprocessing							🟩													
Yuqi Shi	Choosing model and analyze pros and cons										🟩										

PERSON (S)	TASK (S)	Wk5				Wk6				Wk7				Wk8				Wk9			
		MAR								APR											
		S	M	W	R	S	M	W	R	S	M	W	R	S	M	W	R	S	M	W	R
		20	21	23	24	27	28	30	31	3	4	6	7	10	11	13	14	17	18	20	21
Xiaoyan Su Yuqi Shi Jinwei Bi	Build and train models (LSTM, RNN, etc)Build and train models (LSTM, RNN, etc)				🟩																
Xiaoyan Su Yuqi Shi Jinwei Bi	Build and train pre-trained models (BERT, roBERTa, etc)Build and train pre-trained models (BERT, roBERTa, etc)						🟩														
Yuqi ShiYuqi	Model evaluation and comparison											🟩									
Xiaoyan Su Yuqi Shi Jinwei Bi	Wrapping up project report																	🟩			