Hemingway and Carroll: Sentiment Analysis

1. **Description of the cleaning and preprocessing**

   a. Purpose:

In Our Time by Ernest Hemingway and Alice's Adventures in the Wonderland by Lewis Carroll, the novels from homework one and two are evaluated in homework 3 for sentiment analysis. Both novels are labeled as Hemingway and Carroll for ease of discussion in this assignment.

   b. Cleaning process:

In Our Time by Ernest Hemingway and Alice's Adventures in the Wonderland by Lewis Carroll, the novels from homework one are further evaluated in homework three. The purpose of homework three is to conduct sentiment analysis by using train and test datasets. The training dataset is a corpus with sentiment related to Airline Services and saved as train_tweets_airline in a CSV file format. Within the training dataset, there are 15 columns. Still, only two essential columns will be applied for the analysis are text and airline_sentiment. The text consists of customer's Twitter comments on their experience with a specific airline. The airline_sentiment column labeled the customer's text as 'positive,' 'negative,' or 'neutral.' The Hemingway and Carroll text files are converted to CSV files specifically as test datasets for sentiment analysis. In each test dataset, the first row must label as a text for data processing.

A data cleaning function using a regular expression to remove "@" and "#" from the training set and "n't" and "'s" from Hemingway's test set. However, the removal of "n't" and "'s" from Hemingway's test set decreased the number of positive and negative. Therefore, no data cleaning was performed in both test sets.

```
def remove_at(x):
    x = str(x).replace('@', '')
    x = str(x).replace('#', '')
    return x
```

Figure 1: Regular Expression for Cleaning

c. Tokenize

The training set is tokenized using the RegexpTokenizer (figure 2) to select the only word by the '\w+' and eliminate unnecessary Twitter noises. Then, identify and tally which sentences are considered positive, negative, or neutral (Figure 3). The airline_sentiment analysis indicated 9178 negative sentences, 3099 neutral sentences, and 2363 positive sentences, which considered an unbalanced dataset. Since this is the first attempt at using the Twitter training set for Hemingway and Carroll datasets would be beneficial to access the baseline results with the feature.

```
tokenizer = nltk.RegexpTokenizer('\w+')
doc = train_dataset['text'].apply(lambda x : tokenizer.tokenize(x))
```

Figure 2: Tokenize the train_dataset

```
PosSentences = train_dataset[train_dataset['airline_sentiment'] == 'positive']
NegSentences = train_dataset[train_dataset['airline_sentiment'] == 'negative']
NeuSentences = train_dataset[train_dataset['airline_sentiment'] == 'neutral']
```

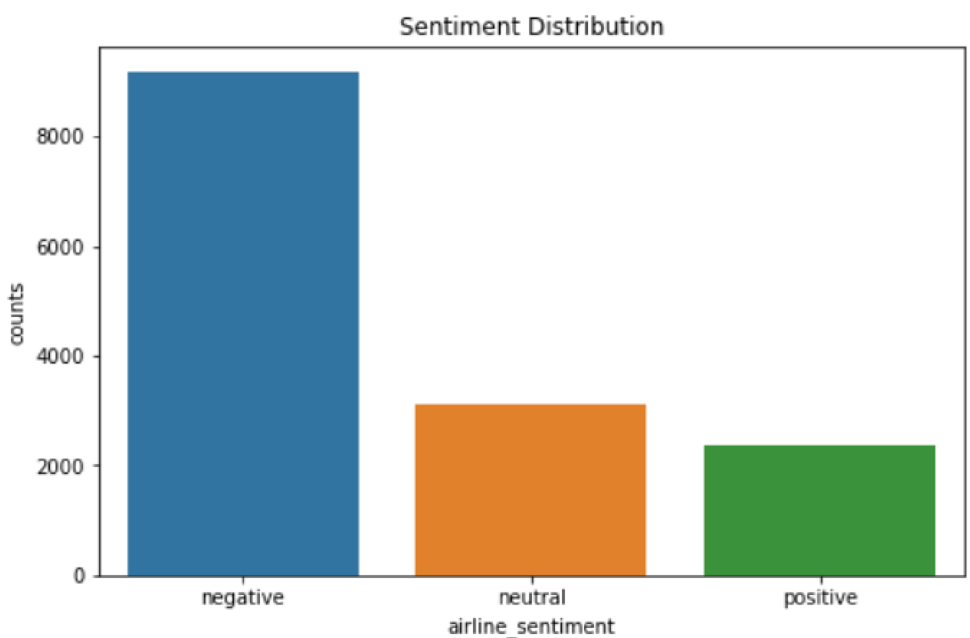

Figure 3: Count of Positive, Negative, and Neutral Sentences from airline_sentiment

2. Classification and Feature Sets

The tokenized sentence is converted to a Python list with its corresponding sentiment label to incorporate into the feature as "docs."

```
for i in range(0, len(train_dataset['airline_sentiment'])):
    # appending the info to the list
    docs.append((doc[i], train_dataset['airline_sentiment'][i]))
```

Figure 4: Tokenize the docs

a. Featuresets: word_features

Randomize the list to eliminate sampling biases before defining the words used for the features. `all_words` extracted from the docs, `top_words` identified the most frequently `all_words` used, and then `most_common_words` listed the top 2000 words in the "docs." Lastly, the `word_features` (Figure 5) is based on the `most_common_words` and used those 2000 words as an essential feature of the "docs."

```
all_words = [word for (sentance,category) in docs for word in sentance]
top_words = nltk.FreqDist(all_words)
most_common_words = top_words.most_common(2000)
word_features = [word for (word,count) in most_common_words]
```

Figure 5: word_features

A function (Figure 6) is defined using the word_features to screen the sentences in the training dataset against the list of most frequent words. The function will be using Boolean logic on each Twitter sentence to check present as 'True' and not present as 'False.'

```
def document_features(document, word_features):
    document_words = set(document)
    #we open a Pytnon dictionary instead of a list
    features = {}
    for word in word_features:
        #checking if the word from word_features matches a word in the document
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

Figure 6: Word_feature function

The `featuresets` (Figure 7) is set for the classification by using the word_feature function for the train and test datasets.

```
featuresets = [(document_features(d, word_features), c) for (d, c) in docs]
```

Figure 7: Featuresets

b. Featuresets2: bigram_features

Nltk.collocations and re are imported for building featuresets2. Finder pulled the words from the `all_words`, then filtered the words by alpha and stopwords. Then `scored` listed the top word pairs based on the `bigram_measures`. Then `bigram_features` (Figure 8) based on the `scored` to use those 2000 word pairs as an essential feature of the "docs."

```
bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(all_words)
finder.apply_word_filter(alpha)
finder.apply_word_filter(lambda w: w in stopwords)
scored = finder.score_ngrams(bigram_measures.raw_freq)
```

```
bigram_features = [bigram for (bigram, count) in scored[:2000]]
#printing the first 30 for confirmation
bigram_features[:30]
```

Figure 8: Bigram_features

Similar to the word_features function, this function (Figure 9) is defined using the bigram_features to screen the sentences in the training dataset against the list of most frequent word pairs. The function will be using Boolean logic on each Twitter sentence to check present as 'True' and not present as 'False.'

```
def bi_document_features(document, bigram_features):
    document_words = list(nltk.bigrams(document))
    features = {}
    for word in bigram_features:
        #boolean logic will retunt 'True' if there is a match, or 'False' if not
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

Figure 9: Bigram_feauture function

The `featuresets2` (Figure 10) is set for the classification by using the bigram_feature function for the train and test datasets.

```
featuresets2 = [(bi_document_features(d, bigram_features), c) for (d, c) in docs]
```

Figure 10: Featuresets2

c. Classifier and Accuracy Score

Numpy is imported for machine learning using Naive Bayes classifier with 5-fold cross-validation on the featuresets and featuresets2 to train sentiments using word_features and bigram_features to predict sentiment and accuracy scores (Figure 11). Featuresets using word_feature has an accuracy score of 0.769, and featuresets2 using bigram_feature has an accuracy score of 0.679. Based on the accuracy score results, for homework 3 is recommended to use featuresets with word_feature for the Hemingway and Carroll test datasets.

```
import numpy as np
from sklearn.model_selection import KFold

kf = KFold(n_splits = 5)
sum = 0

for train, test in kf.split(featuresets):
    train_data = np.array(featuresets)[train]
    test_data = np.array(featuresets)[test]
    classifier = nltk.NaiveBayesClassifier.train(train_data)
    sum += nltk.classify.accuracy(classifier, test_data)

#storing the score in a variable
acc1 = sum/5
```

Figure 11: Naïve Bayes Classifer for featuresets and feauturesets2

3. Sentiment Analysis on the Test Datasets

a. Test Dataset

Word_feature classifier (Figure 12) is used for the Hemingway and Carroll test datasets, labeled as "text." For this homework assignment, the whole text is processed instead of the recommended ¼ of the text.

```
#iterating over the test file of tweets we crated at the very beginning of the notebook
#PLEASE EDIT THE NUMBER OF DOCUMENTS AS MENTIONED ABOVE
for i in range(0, int(len(test_set['text'])/1)):
    #extracting the text
    sentences = nltk.sent_tokenize(test_set['text'][i])
    #opening the counter to add up positive, negative, or neutral according to predicted labels
    pos_count = 0
    neg_count = 0
    neu_count = 0
    #using our first classifier, the one trained with unigram features
    for sents in sentences:
        senti = classifier.classify(document_features(nltk.word_tokenize(sents), word_features))
        #adding items to the counter as they are classified
        if senti == 'positive':
            pos_sent.append(sents)
            pos_count += 1

        elif senti == 'negative':
            neg_sent.append(sents)
            neg_count += 1

        else:
            neu_sent.append(sents)
            neu_count += 1
    #appending the information from each sentence to the corresponding list
    total_pos.append(pos_count)
    total_neg.append(neg_count)
    total_neu.append(neu_count)
```

Figure 12: Word_feature classifier

The unbalanced data from Twitter would expect more negative sentiment sentences from both test datasets (Figure 13). Still, the test datasets did not reflect that. The classifier from the Carroll test dataset generated 1296 positive, 588 negatives, and 1094 neutral sentiment sentences. The Hemingway test dataset has generated 3329 positive, 400 negatives, and 1934 neutral sentiment sentences. Unexpectedly, the Hemingway neutral and positive sentiment sentences were higher than Carroll by observing the raw outputs. Normalized results (Figure 14) showed that Hemingway has higher positive and lower negative sentiment sentences than Carroll. This finding did not expect Hemingway to be an optimistic writer than Carroll, or the training dataset needed to be balanced for the second round of machine training.

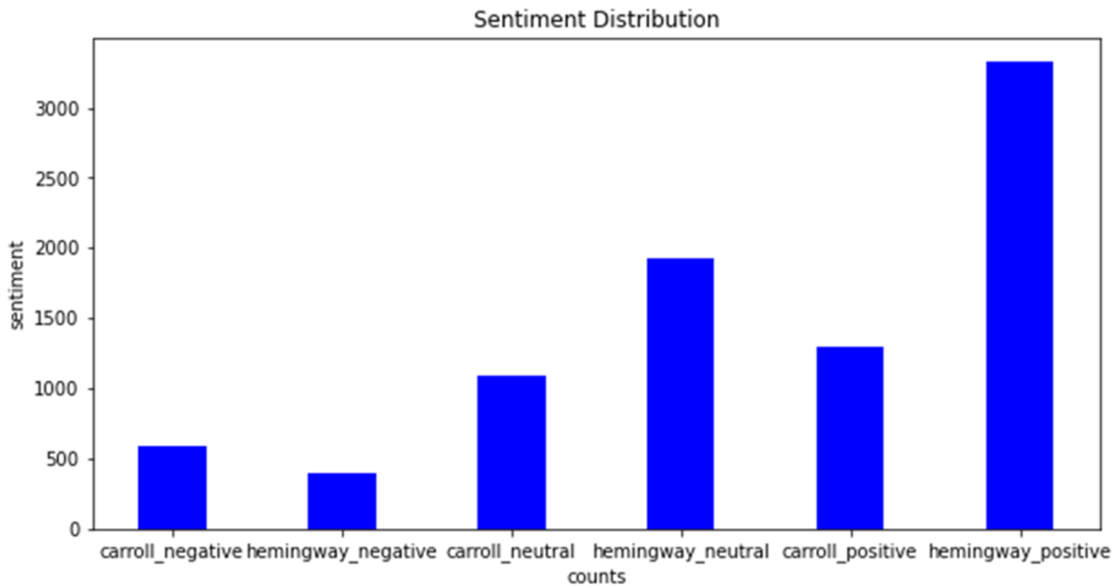

Figure 13: Test Datasets Sentiment Distribution

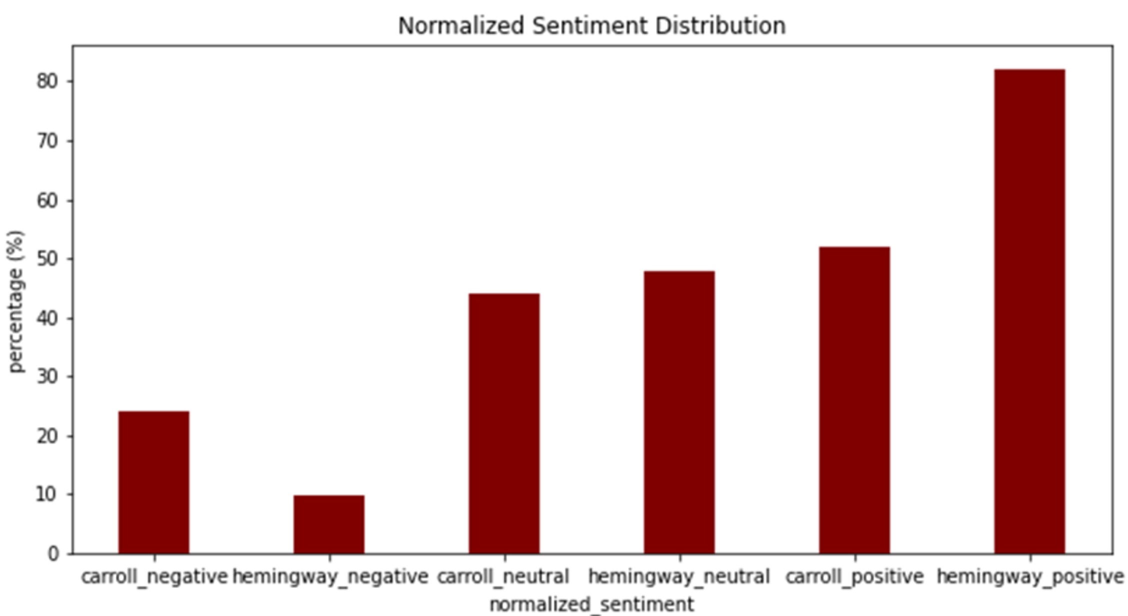

Figure 14: Test Datasets Normalized Sentiment Distribution

b. Grammar_phrases function

The grammar_phrases function is created to extract top 50 adjective phrases, adverb phrases, and verb phrases (Figure 15). Different grammar_phrases were used for adjective, adverb, and verb phrases as shown below.

```
grammar_adjph = "ADJPH: {<RB.?>+<JJ.?>}"
```

```
grammar_advph = "ADVPH: {<RB>+<RB>}"
```

```
grammar_vbph = "VBPH: {<VB.?>+<VB.?>}"
```

```
def grammar_phrases(tags_sent):
    grammar_adjph = "ADJPH: {<RB.?>+<JJ.?>}" # REMEMBER TO EDI THIS GRAMMAR FOR ADVERBS & VERBS!
    chunk_parser_adj = nltk.RegexpParser(grammar_adjph)
    adjph_tags = []
    for sent in tags_sent:
        if len(sent) > 0:
            tree = chunk_parser_adj.parse(sent)
            for subtree in tree.subtrees():
                if subtree.label() == 'ADJPH': # THIS ALSO NEEDS EDITION FOR ADVERBS & VERBS
                    adjph_tags.append(subtree)
    return adjph_tags


# EXTRACTING ADJECTIVE PHRASES without POS tags, just the phrase
# we also reuse this but replacing the 'tagged_phrase' in each new case
def word_phrase(tagged_phrase):
    adjective_phrases = []
    for sent in tagged_phrase:
        temp = ''
        for w, t in sent:
            temp += w+ ' '
        adjective_phrases.append(temp)
    return adjective_phrases


# RANKING BY FREQUENCY
# this is also a function to reuse
def get_frequency(phrases):
    phrases_frequency = nltk.FreqDist(phrases)
    return phrases_frequency.most_common(50)
```

Figure 15: Grammar_phrases function

Also, this function can be used with "tags_pos", "tags_neg" or "tags_neu" to identify positive, negative, and neutral phrases (Figure 16). The acronym for the grammar changed from adjph to advph or vbph.

Lastly, Figures 17 and 18 show the TOP 50 negative and positive adjectives, adverb, and verb phrases.

```
# EXTRACTING POSITIVE PHRASES AND THEIR POS
adjph_pos = grammar_phrases(tags_pos)
print('Adjective phrases in positive sentences, with POS: ', adjph_pos[:3])

# EXTRACTING POSITIVE ADJECTIVE PHRASES (WORDS ONLY)
word_adjph_pos = word_phrase(adjph_pos)
print('First 10 adjective phrases in positive sentences: ', word_adjph_pos[:10])

# RANKING POSITIVE PHRASES BY FREQUENCY
most_common_adjph_pos = get_frequency(word_adjph_pos)
print("Top 50 adjective phrases in positive sentences: ", most_common_adjph_pos[:50])
```

Figure 16: Tags

Conclusion

Surprisingly, not many negative sentiment words from the Hemingway and Carroll test datasets were identified using an unbalanced training dataset. Instead, more positive sentiment words were identified. The sentiment analysis indicated that Hemingway as a writer might be more optimistic than Carroll. Also, confirmed on the finding from homework 2 that Hemingway used more adverbs than adjectives in his writing. Originally, Carroll was thought to be a whimsical writer. Still, these findings indicated that he might be a less optimistic writer than Hemingway. Or Carroll might be a sarcastic writer.

	adjective phrase_pos	adjective phrase_neg	adverb phrase_pos	verb phrase_pos	verb phrase_neg	adverb phrase_neg
0	(so much , 5)	(so much , 3)	(as well , 10)	(had been , 6)	(was going , 6)	(down again , 2)
1	(very curious , 4)	(very little , 2)	(very soon , 5)	(had made , 4)	(had been , 5)	(very politely , 2)
2	(very much , 3)	(very glad , 2)	(down here , 3)	('ve seen , 4)	(was sitting , 5)	(so far , 2)
3	(very few , 2)	(n't very civil , 2)	(As soon , 3)	(was talking , 3)	(have been , 4)	(n't very , 2)
4	(very good , 2)	(very tired , 1)	(very nearly , 3)	(was getting , 3)	(was beginning , 3)	(just now , 2)
5	(always ready , 2)	(so _very_ , 1)	(just as well , 3)	(was looking , 3)	(had got , 3)	(never before , 1)
6	(very difficult , 2)	(no longer , 1)	(very much , 3)	('ve tried , 3)	('ve got , 2)	(not here , 1)
7	(quite silent , 2)	(almost certain , 1)	(very slowly , 2)	(had got , 2)	(be lost , 2)	(too long , 1)
8	(so _very_ , 2)	(very nice , 1)	(very earnestly , 2)	(were getting , 2)	('s getting , 2)	(not possibly , 1)
9	(not much , 2)	(very fond , 1)	(very well , 2)	(was surprised , 2)	(has won , 2)	(rather sharply , 1)
10	(too much , 2)	(so small , 1)	(too far , 2)	(had come , 2)	(was delighted , 2)	(up again , 1)
11	(very glad , 2)	(so out-of-the-way , 1)	(as hard , 2)	(was pressed , 2)	('ve tried , 2)	(alone here , 1)
12	(very interesting , 2)	(very clear , 1)	(very politely , 2)	('ve got , 2)	(had gone , 2)	(as nearly , 1)
13	(once more , 2)	(not easy , 1)	(very gravely , 2)	(have got , 2)	(was gone , 2)	(away altogether , 1)
14	(very sleepy , 1)	(quite dry , 1)	(very carefully , 2)	(was sneezing , 2)	(was looking , 2)	(long ago , 1)
15	(so _very_ much , 1)	(very absurd , 1)	(never even , 2)	('re doing , 2)	(be raving , 2)	(slowly back , 1)
16	(quite natural , 1)	(so grave , 1)	(as long , 2)	(were learning , 2)	('ve had , 2)	(well enough , 1)
17	(very deep , 1)	(here poor , 1)	(so often , 2)	(was gone , 2)	(were trying , 2)	(very soon , 1)
18	(very likely , 1)	(very uncomfortable , 1)	(so _very_ , 1)	(was considering , 1)	(was obliged , 2)	(not even , 1)
19	(too large , 1)	(almost wish , 1)	(deep well , 1)	(were filled , 1)	(had peeped , 1)	(now here , 1)
20	(too small , 1)	(together first , 1)	(not even , 1)	(was labelled , 1)	(was reading , 1)	(enough yet , 1)
21	(much larger , 1)	(enough yet " , 1)	(so Alice , 1)	(was dozing , 1)	(be seen , 1)	(about again , 1)
22	(really impossible , 1)	(very likely , 1)	(now only , 1)	(had begun , 1)	(was lit , 1)	(not long , 1)
23	(now only ten , 1)	(quite tired , 1)	(quite plainly , 1)	(having seen , 1)	(thought was , 1)	(back again , 1)
24	(too slippery , 1)	(so many , 1)	(very seldom , 1)	(be ashamed , 1)	(had taught , 1)	(as well wait , 1)
25	(very small , 1)	(too weak , 1)	(so far , 1)	(came trotting , 1)	(be shutting , 1)	(nearly as , 1)
26	(quite surprised , 1)	(more subdued , 1)	(Just then , 1)	('ve been changed , 1)	(had forgotten , 1)	(very well , 1)
27	(quite dull , 1)	(more puzzled , 1)	(ever so , 1)	(is twelve , 1)	(had tired , 1)	(very much , 1)
28	(now more , 1)	(very sorry , 1)	(perhaps not , 1)	(were saying , 1)	(remembered trying , 1)	(not so , 1)
29	(very hot , 1)	(so long , 1)	(so nicely , 1)	(was shut , 1)	(have been changed , 1)	(even then , 1)
30	(very little , 1)	(nearly as large , 1)	(n't indeed , 1)	(being drowned , 1)	(saying "Come , 1)	(very readily , 1)
31	(ever so many , 1)	(quite impossible , 1)	(rather crossly , 1)	(_will_ be , 1)	(like being , 1)	(asleep again , 1)
32	(too bad , 1)	(as much , 1)	(very humbly , 1)	(remembered having seen , 1)	(had put , 1)	(very angrily , 1)
33	(curly brown , 1)	(neither more , 1)	(very angrily , 1)	(sits purring , 1)	(was holding , 1)	(very humbly , 1)
34	(as hard , 1)	(quite absurd , 1)	(so easily , 1)	(was bristling , 1)	(avoid shrinking , 1)	(_there_ again , 1)
35	(more energetic , 1)	(not so mad , 1)	(as soon , 1)	(was trembling , 1)	(was lying , 1)	_You'd_ better not , 1)
36	(soon left , 1)	(so large , 1)	(slowly back again , 1)	('ve offended , 1)	(had fallen , 1)	(angrily away , 1)
37	(as sure , 1)	(so confused , 1)	(very good-naturedly , 1)	(was swimming , 1)	(be said , 1)	(then quietly , 1)
38	(much sooner , 1)	(Once more , 1)	(Very soon , 1)	(had known , 1)	(do it.□ , 1)	(down again very , 1)
39	(quite enough , 1)	(rather doubtful , 1)	(so indeed , 1)	(had finished , 1)	(were placed , 1)	(certainly not , 1)
40	(quite so much , 1)	(very uneasy , 1)	(quite so , 1)	(be offended , 1)	(had been running , 1)	(up very sulkily , 1)
41	(too late , 1)	(very hopeful , 1)	(better now , 1)	(began wrapping , 1)	(was speaking , 1)	(now hastily , 1)
42	(very confusing.□ , 1)	(only wish , 1)	(very neatly , 1)	(be getting , 1)	(had changed , 1)	(perhaps even , 1)
43	(very melancholy , 1)	(rather better , 1)	(so yet , 1)	(began hunting , 1)	(had lost , 1)	
44	(very white , 1)	("The more , 1)	(very queer , 1)	(be seen , 1)	(have dropped , 1)	
45	(perfectly sure , 1)	(simply " , 1)	(back again , 1)	(have changed , 1)	(be turned , 1)	
46	(very supple , 1)	(quite unhappy , 1)	(so awfully , 1)	(had vanished , 1)	(had found , 1)	
47	(as steady , 1)	(so ordered , 1)	(rather doubtfully , 1)	(went hunting , 1)	(had hoped , 1)	
48	(very sudden , 1)	(very long , 1)	(down again , 1)	(better take , 1)	(being broken , 1)	
49	(very truthful , 1)	(too close , 1)	(certainly there , 1)	(had found , 1)	(be afraid , 1)	

Figure 17: Carroll Top 50 Positive and Negative Adjective, Adverb, and Verb Phrases

	adjective phrase_pos	adverb phrase_pos	verb phrase_pos	verb phrase_neg
0	(too much , 3)	(n't ever , 3)	(had been , 12)	(had been , 6)
1	('Do many , 2)	(as far , 2)	('ve got , 9)	(was gone , 4)
2	(so much , 2)	(far away , 2)	(was looking , 5)	(is going , 3)
3	(absolutely perfect , 2)	(n't much , 2)	(was excited , 5)	(had gone , 2)
4	(pretty good , 2)	(n't so , 2)	(had made , 4)	(was sitting , 2)
5	(very fine , 2)	(very hard , 1)	(was sitting , 3)	(had done , 2)
6	(n't engaged/ , 2)	(farther ahead , 1)	(was gone , 3)	(had been trying , 1)
7	(too many , 2)	(very badly , 1)	('s got , 3)	(is called being , 1)
8	(too big , 2)	(Just then , 1)	(be married , 3)	(had been gone , 1)
9	(so big , 2)	(very carefully , 1)	(had come , 3)	(had been lost , 1)
10	(very big , 1)	(rather not , 1)	(was going , 3)	(had drifted , 1)
11	(very bad , 1)	(away so , 1)	(was getting , 3)	(were done , 1)
12	(very pale , 1)	(pretty quietly , 1)	(was drunk , 2)	(was going , 1)
13	(away wet , 1)	('Hardly ever , 1)	(were going , 2)	(stood stacked , 1)
14	(terribly sorry , 1)	(once again , 1)	(was working , 2)	(got going , 1)
15	(very exceptional , 1)	(not quite , 1)	(was smoking , 2)	(had put , 1)
16	(very many , 1)	(sore as , 1)	(was heating , 2)	(understand was , 1)
17	(very lazy , 1)	(not even very , 1)	(had brought , 2)	('s missed , 1)
18	(very serious , 1)	(very quietly , 1)	(were jammed , 2)	(sat looking , 1)
19	(awfully surprised , 1)	(Right away , 1)	(were trolling , 2)	(had lost , 1)
20	(afrightfully hot , 1)	('All right , 1)	(was blowing , 2)	(had felt , 1)
21	(simply priceless , 1)	(back again , 1)	(had fallen , 2)	(had swung , 1)
22	('How much , 1)	(ahead brilliantly , 1)	(had gotten , 2)	(had said , 1)
23	(consciously practical , 1)	(back much too , 1)	(was made , 2)	(had spoken , 1)
24	(quite proud , 1)	(back so , 1)	(had heard , 2)	(being written , 1)
25	(thoroughly practical , 1)	(not very , 1)	(had grown , 2)	(have made , 1)
26	(awfully big , 1)	(ever again , 1)	(was listening , 2)	(had dragged , 1)
27	(very wise , 1)	(Not now , 1)	(came pounding , 2)	(was missing , 1)
28	(n't drunk , 1)	(so much , 1)	(have been , 2)	(be listened , 1)
29	(really drunk , 1)	(only too , 1)	(were rising , 2)	(had told , 1)
30	(not even very important , 1)	(down beside , 1)	(was done , 2)	(was changed , 1)
31	(very flat , 1)	(all right , 1)	(had taken , 2)	(has felt , 1)
32	(not beautiful , 1)	(over now , 1)	(was starting , 2)	(gotten going , 1)
33	(back much too late , 1)	(Not exactly , 1)	(was cold , 1)	('s got , 1)
34	(back so late , 1)	(too far away , 1)	(was rowing , 1)	(get married , 1)
35	(quite unimportant , 1)	(not myfather , 1)	(was soaking , 1)	(go skiing , 1)
36	(so many , 1)	(along back , 1)	(was cut , 1)	(was crouched , 1)
37	(n't worth , 1)	(always not , 1)	(had been helping , 1)	(was riding , 1)
38	(n't true , 1)	(Then once , 1)	(had cut , 1)	(be jogging , 1)
39	(really ripe , 1)	(never really , 1)	(was satisfied , 1)	(had bumped , 1)
40	(not worth , 1)	(n't really , 1)	(had rowed , 1)	(was getting , 1)
41	(most interesting , 1)	(well back , 1)	(was doing , 1)	(was scared , 1)
42	(really good , 1)	(farther away , 1)	(were closed , 1)	(had happened , 1)
43	(n't much good , 1)	(about twice , 1)	(was feeling exalted , 1)	(do straighten , 1)
44	(only too pleased , 1)	(out again , 1)	(was standing , 1)	(getting dressed , 1)
45	(just angry , 1)	(n't hardly , 1)	(had been cut , 1)	(was looking , 1)
46	(more scene , 1)	(As soon , 1)	(had flowed , 1)	(have thought , 1)
47	(straight up-hill , 1)	(far down , 1)	(were seated , 1)	(was leading , 1)
48	(n't so cordial , 1)	(much too , 1)	(was coming , 1)	(were piled , 1)
49	(too rocky , 1)	(too steadily , 1)	(were hauling , 1)	(went slamming , 1)

	adjective phrase_neg
0	(not worth , 1)
1	(probably bad , 1)
2	(no longer so tragic , 1)
3	(quite close , 1)
4	(not enOtigh , 1)
5	(rather ridiculous , 1)
6	(not sensational , 1)
7	(once more , 1)
8	(n't really bad , 1)
9	(so awfully dead , 1)
10	(too hot , 1)
11	(too slow , 1)
12	(very biggest , 1)
13	(almost impossible , 1)

	adverb phrase_neg
0	(n't ever , 3)
1	(n't really , 2)
2	(here now , 2)
3	(down beside , 1)
4	(too late , 1)
5	('So long , 1)
6	(Outside now , 1)
7	(no longer so , 1)
8	(long back , 1)
9	(not fully , 1)
10	(up again , 1)
11	(sure never , 1)
12	(back again ever , 1)
13	(so awfully , 1)
14	(slowly up , 1)
15	(faster now , 1)
16	(really too , 1)
17	(never still , 1)
18	(away slowly , 1)
19	(not too , 1)

Figure 18: Hemingway Top 50 Positive and Negative Adjective, Adverb, and Verb Phrases