Project Report for Hemingway Next Word Prediction

by Kirk Hazen
Springboard
Data Science Career Track
Mentor: Raghunandan Patthar

ORCID: https://orcid.org/0000-0001-6402-5176

Acknowledgements: I would like to thank my mentor, Raghunandan Patthar, for guidance with this project.

Email: Kirk.Hazen@icloud.com

Ernest Hemingway:
Ernest Hemingway was an influential and popular writer of the twentieth century. As both a journalist and a novelist he captured many readers' attention with his vivid yet simple descriptions of life. With three novels published in his life, three others published posthumously, and a total of 19 short-story collections, plus many non-fiction pieces, he was an influential writer for many readers and fellow writers. His style of prose provides an interesting challenge for next-word-prediction through large language models. For this project, one of his earliest and most highly regarded works, *The Sun Also Rises*, will be used to train a large language model.

The Data:
The novel *The Sun Also Rises* was published in 1926 and the first-edition cover can be seen in Figure 1. It is an early modernist novel that chronicles the adventures and relations of American and British expatriates who drink and carous their way from Paris to Spain to watch the bullfights and live life fully. The novel is still in print, nearly a 100 years later, and its allure stems partly from it being a book based in part on Hemingway's own life and those of his traveling companions in the early 1920s. It is often considered Hemingway's greatest and most important single work.

Large Language Models:
As Kochmar (2022) explains, the history of Natural Language Processing (NLP) is one which has evolved rapidly over the past few decades. Whereas rule-based algorithms formed the foundation for most NLP efforts in the 1970s, the introduction of statistical approaches in the 1980s along with the ever expanding sea of data from the World Wide Web in the 1990s allowed NLP methods to advance rapidly. By the 2010s, deep learning was possible with new developments in hardware. The approach used in this project is based on Google's Bidirectional Encoder Representations from Transformers (BERT), a pretrained model of language knowledge (https://arxiv.org/abs/1810.04805).

Next Word Prediction:
The task of next-word prediction involves assessing the words in a phrase and then making an educated guess as to what the next word would be. Most smart phones have text apps that provide suggested words, and this use of suggested words comes from next word prediction. If you have ever used Google Docs, the autocomplete function that suggests words is based on BERT. It is a statistical process based on the knowledge gathered through the pretraining of the BERT model. To speed along the training, I used the slightly condensed but almost as efficient DistilBERT model from Hugging Face. The open question for this Capstone is whether BERT's predictions can be improved by training BERT on part of a Hemingway novel?

Model Training:
Domain adaptation is the process of fine-tuning a pretrained language model on a specific genre of text or domain of knowledge. If a pretrained model is to be used for work within linguistics, knowing words such *tautosyllabicity* and *monophthongization* would be helpful. The domain adaptation for this project is to fine-tune the pretrained model on Hemingway's writing so that it might better predict the next word in a masked language task.

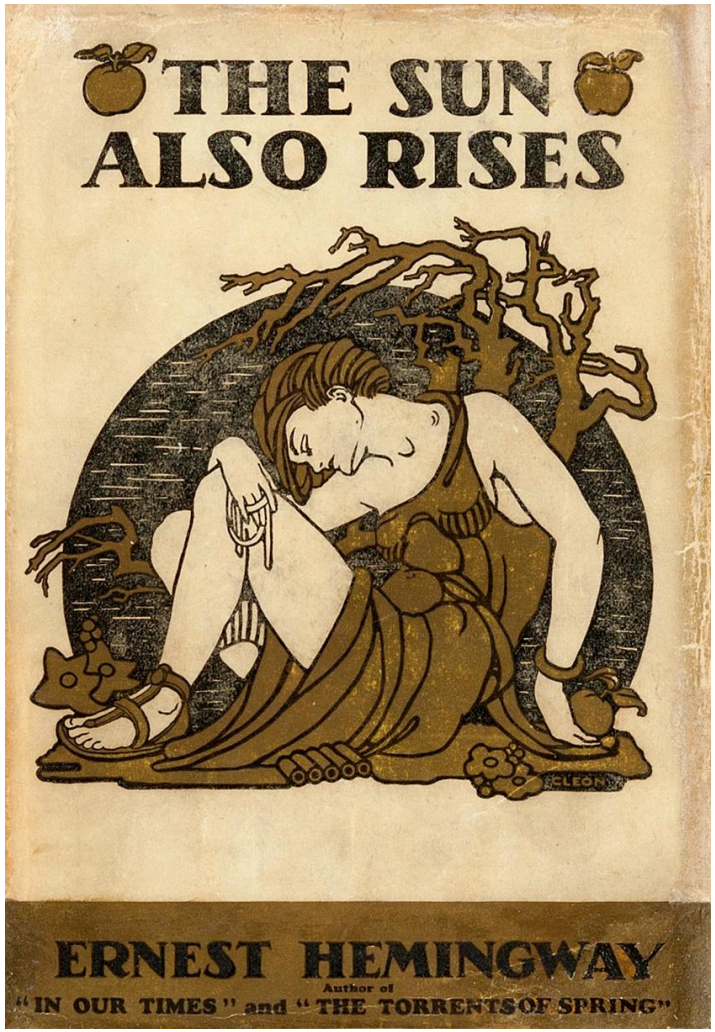


Figure 1: First Edition Cover

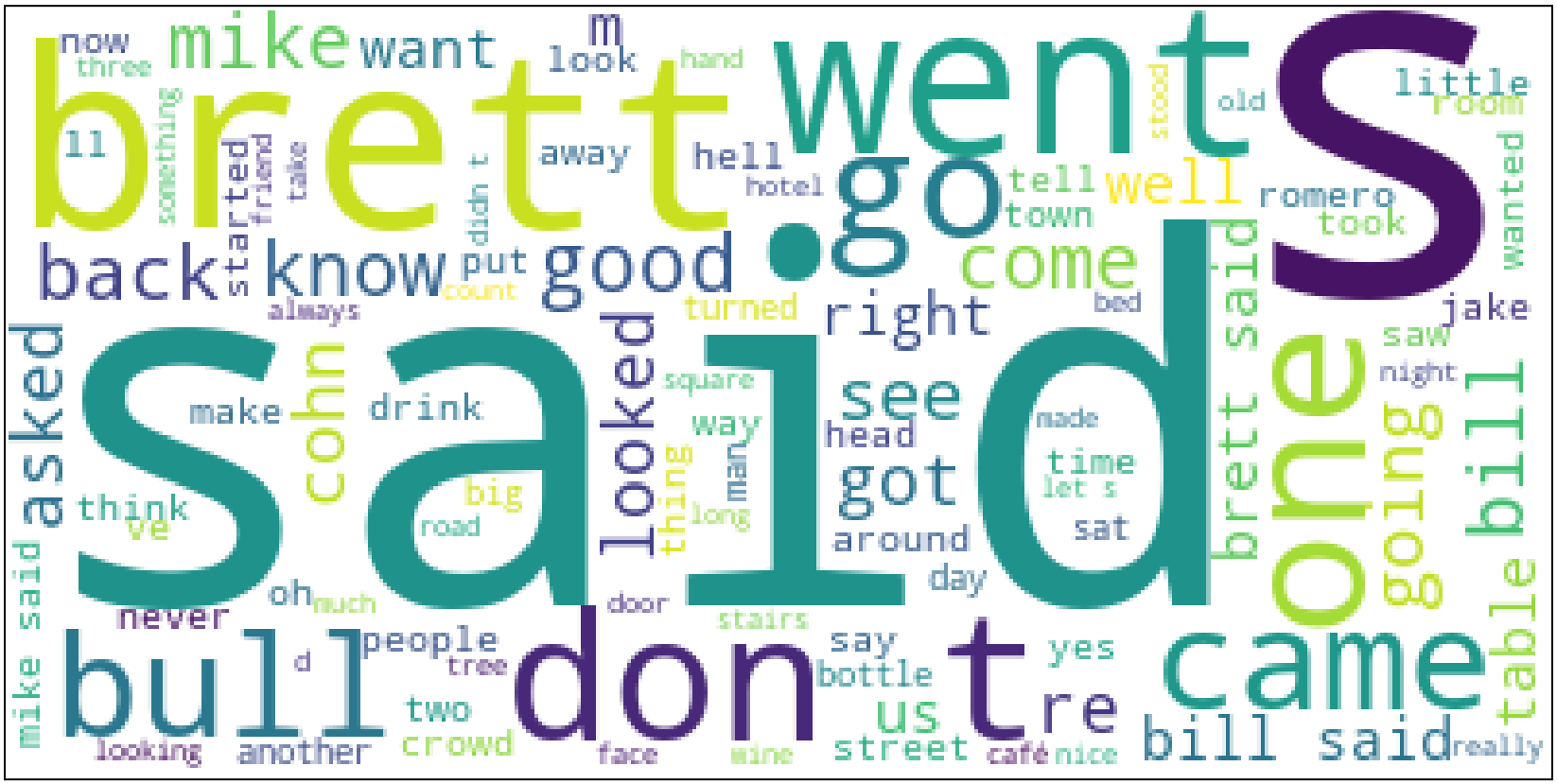
Data Wrangling and Exploratory Data Analysis:

The text of the entire novel was pulled from Project Gutenberg (https://www.gutenberg.org/). After cleaning up the disclaimers and additional materials, the novel itself had 364,517 characters, including spaces. For text based words (not numerals/not punctuation), there are 68,315 . The question of stop words is ever present for data wrangling with NLP projects. Stop words are the most common function word in the English

language, and whether they should be removed depends on the requirements of your project. For this project these were only removed during the phases that required a closer look at the main lexical items for this novel. The NLP packages tokenize and stopwords from nltk were used when stopwords were removed, such as for ngrams. The python translate() method was used to remove punctuation, including an expanded range of single and double quotes.

The Sun Also Rises is largely based on dialogue, and the word cloud in Figure 2 shows that "said" is the single most frequent word in the novel. This word cloud comes from the 100 most frequent words in the novel. Some interesting themes appear in this word cloud. Clearly the characters Brett, Mike, Bill and others were important. People were coming and going (with the explicit *go*, *went*, and *come*).

Figure 2: Word Cloud of the 100 most frequent words in *The Sun Also Rises*



As this word cloud view is after stop words and punctuation were removed, but before clean up of other items like possessive *'s*, which clearly played a prominent role in the novel. There were actually 15,155 of them in the novel. In all, there were 19,371 punctuation marks, 748 numerals, and 68,135 text-based words. An example of the novel dialogue is given in Figure 3.

Figure 3: A sample of text and dialogue from the novel

"Nobody ever lives their life all the way up except bull-fighters."

"I'm not interested in bull-fighters. That's an abnormal life. I want to go back in the country in South America. We could have a great trip."

"Did you ever think about going to British East Africa to shoot?"

"No, I wouldn't like that."

"I'd go there with you."

"No; that doesn't interest me."

"That's because you never read a book about it. Go on and read a book all full of love affairs with the beautiful shiny black princesses."

After exploring various combinations of adjectives and nouns, a general assessment of most common adjectives, nouns, and verbs was taken. Hemingway became famous in part for his sparse style of writing. These graphs also reveal some of the most common themes for the novel.

In Figures 4-6, the counts of the most frequent nouns, verbs, and adjectives are given.

Figure 4: Most frequent nouns in the novel

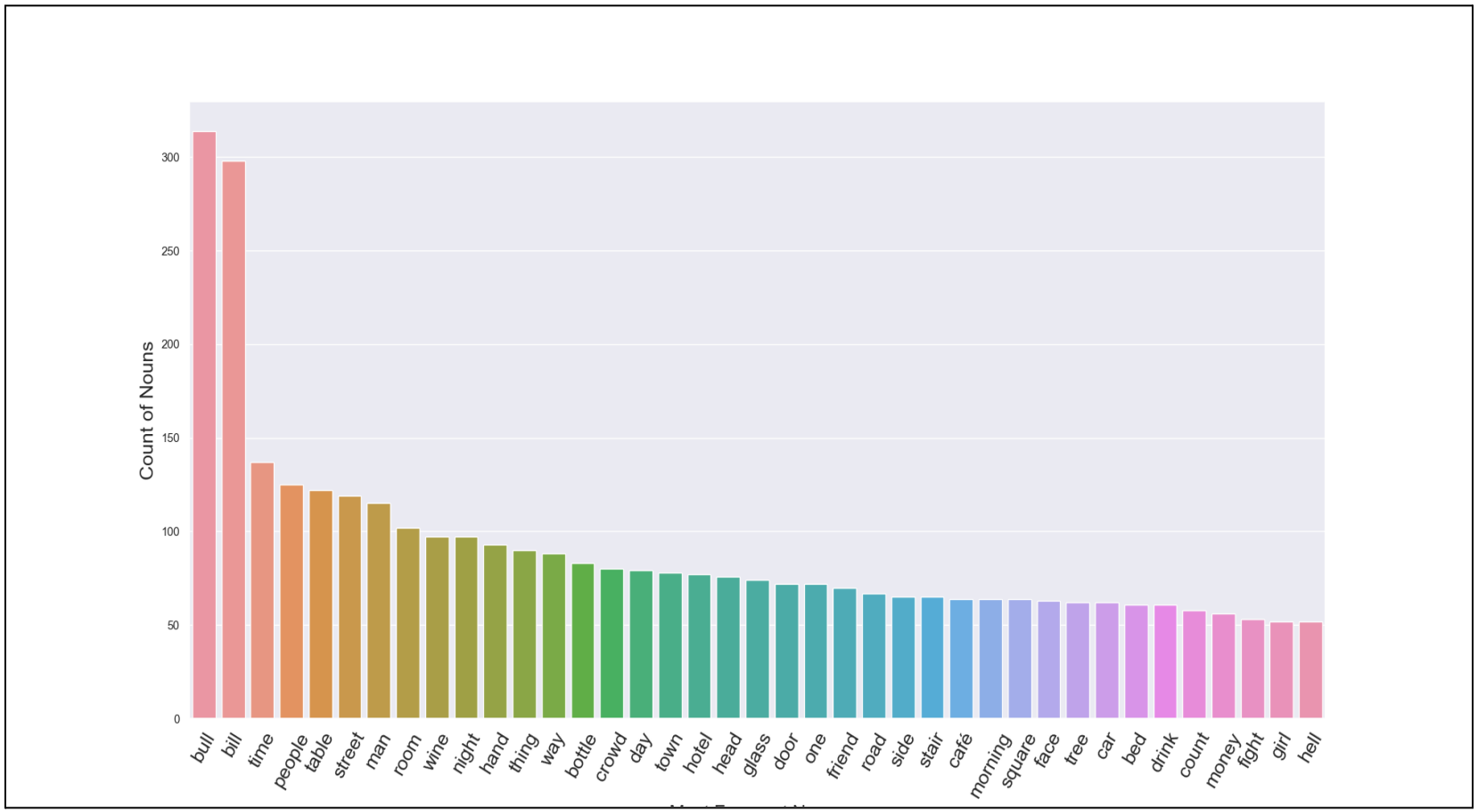


Figure 5: Most frequent verbs in the novel

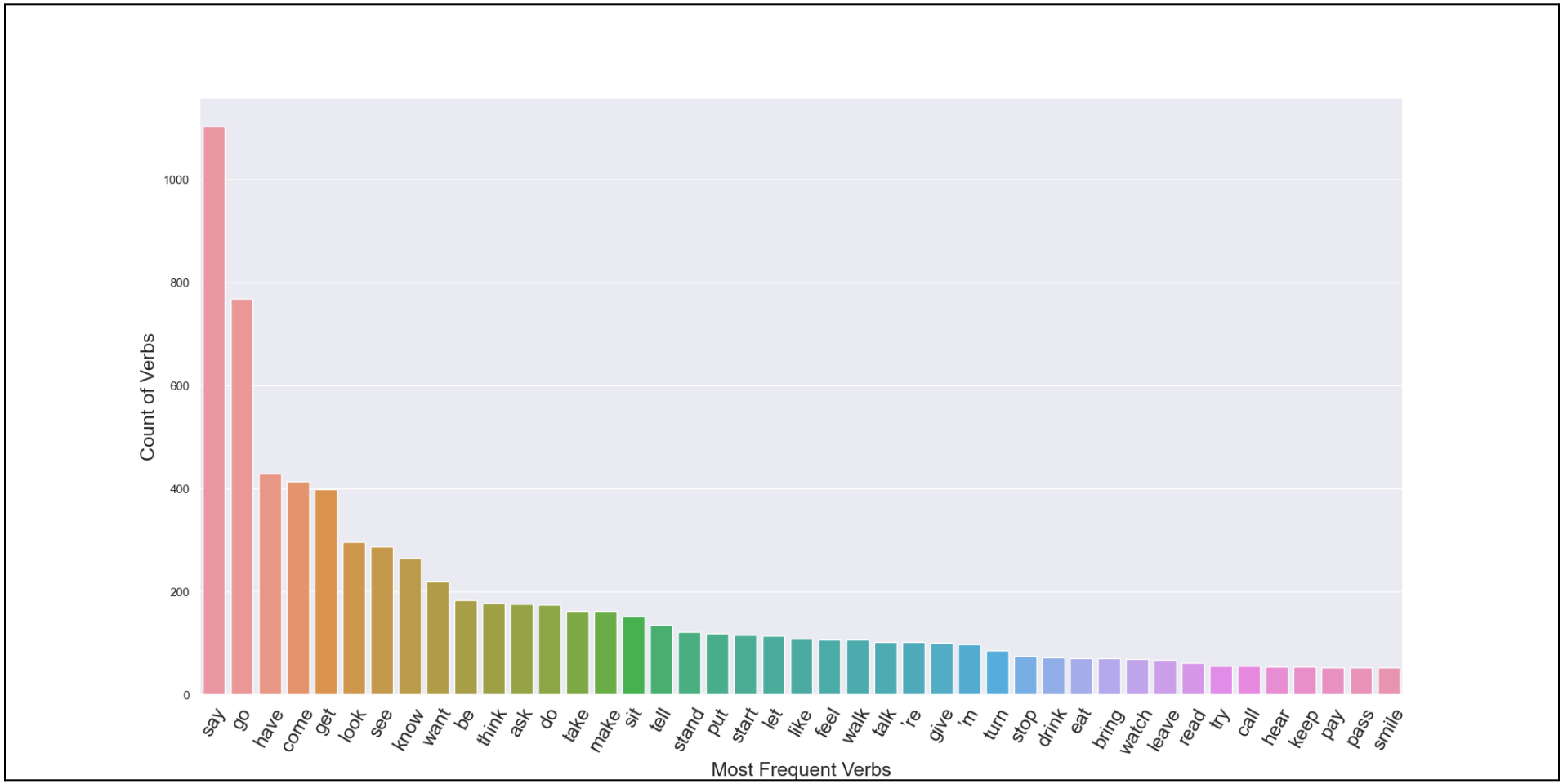
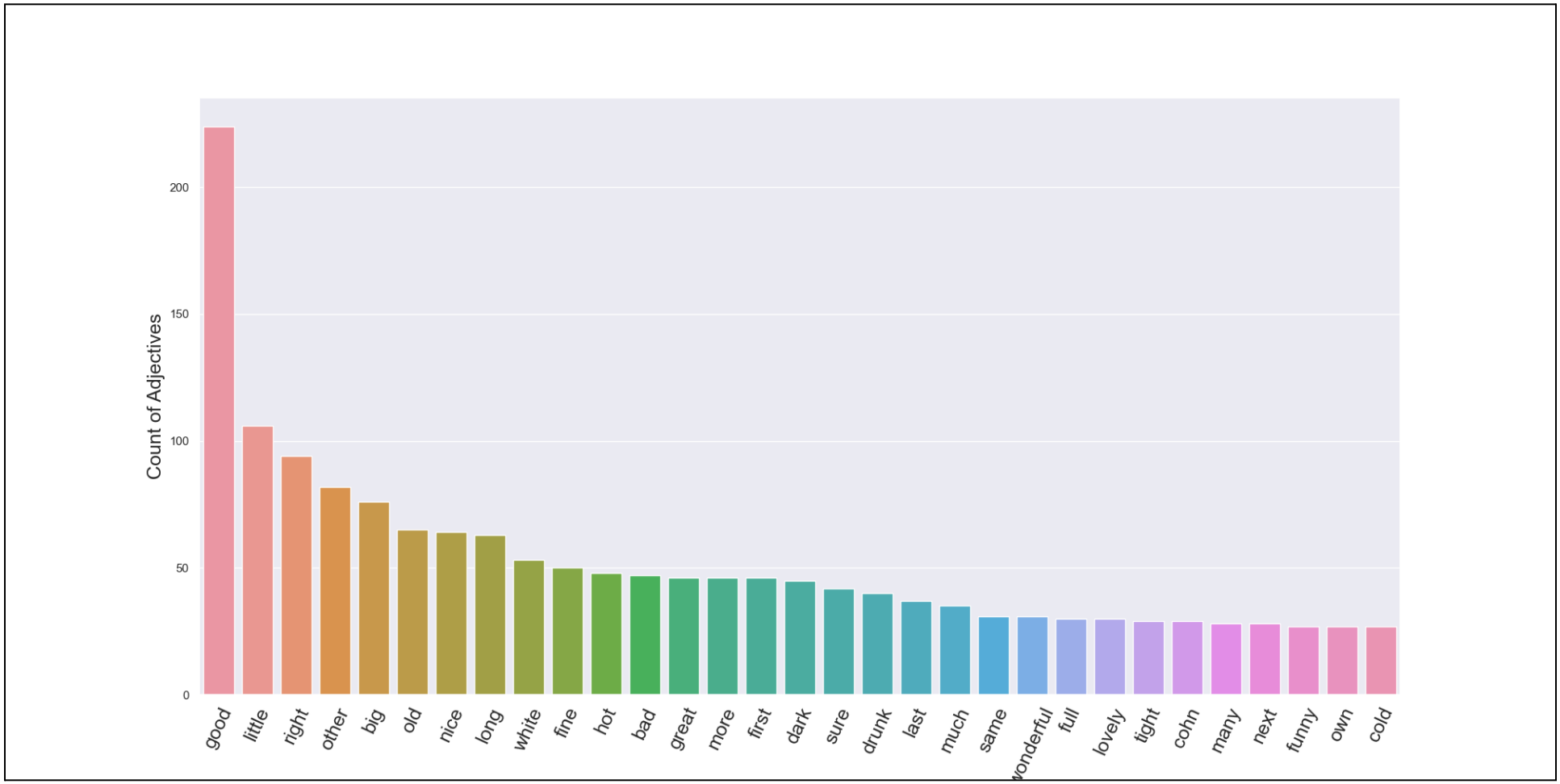


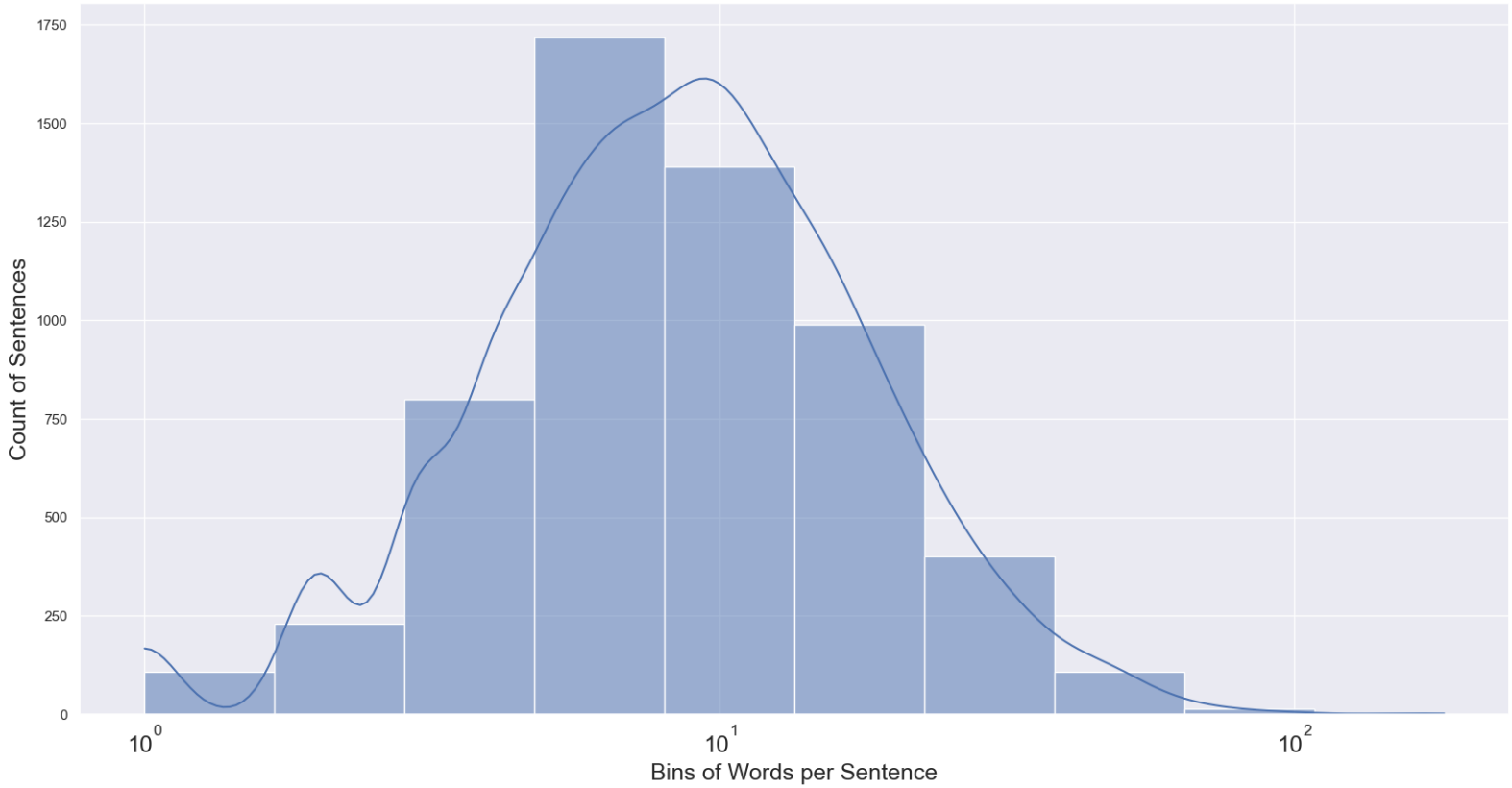
Figure 6: Most frequent adjectives in the novel



One notable thing about all three graphs is that a few items dominate their respective charts. The top five adjectives are *good*, *little*, *right*, *other*, and *big*. The top five verbs are *say*, *go*, *have*, *come*, *get*. The top five nouns are *bull*, *bill*, *time*, *people*, *table*. Yet the most striking observation is that all of these words have been present in English since at least Middle English (1500-1700). This reliance on frequently used basic vocabulary is part of Hemingway's style.

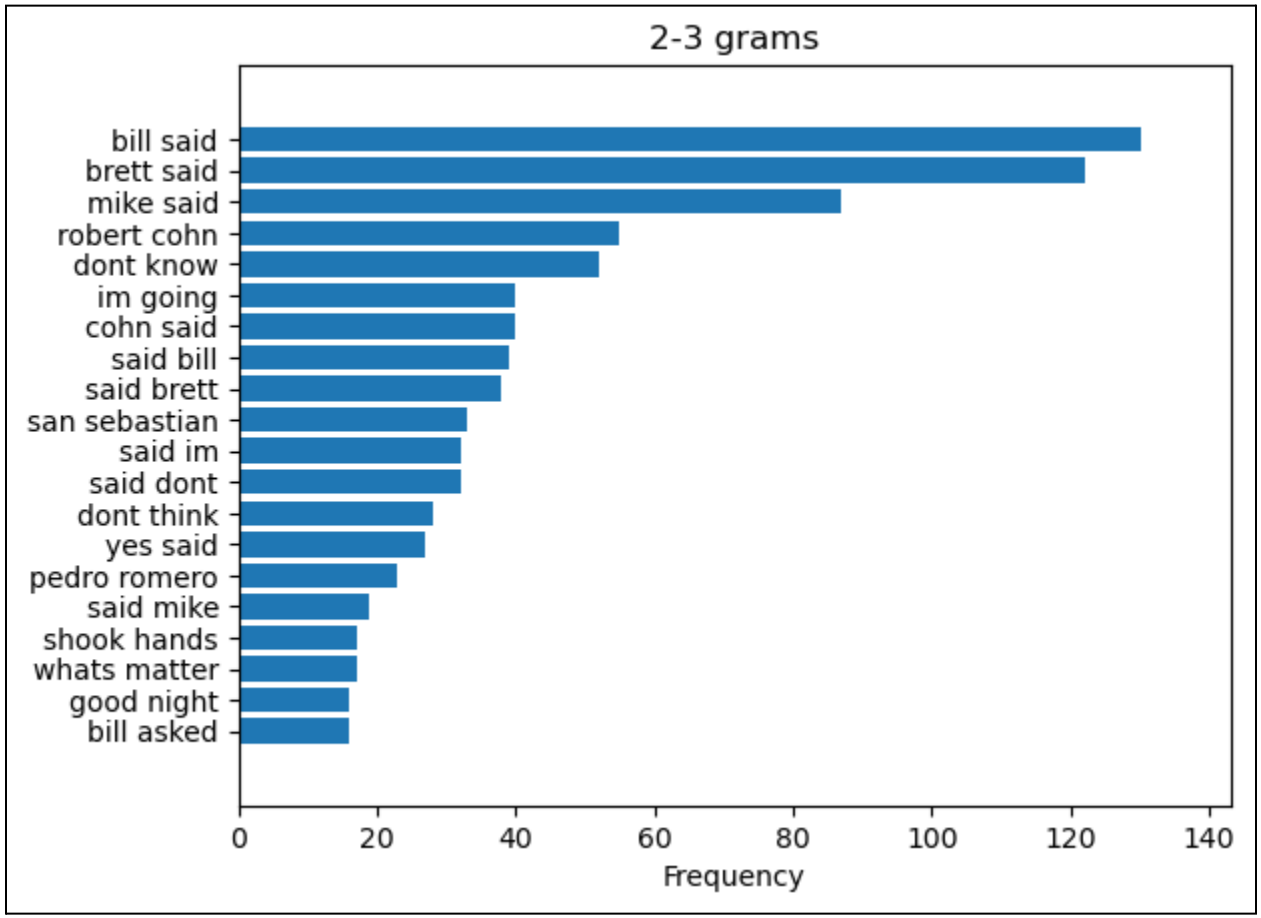
In assessing sentences as a unit, does Hemingway have long sentences in this hallmark novel? A histogram and smoothed trend line show that most sentences have below 20 words per sentence, with most sentences having below 10 words per sentence. Figure 7 shows the log-transformed histogram to allow a more even distribution.

Figure 7: Log-transformed histogram of words per sentence



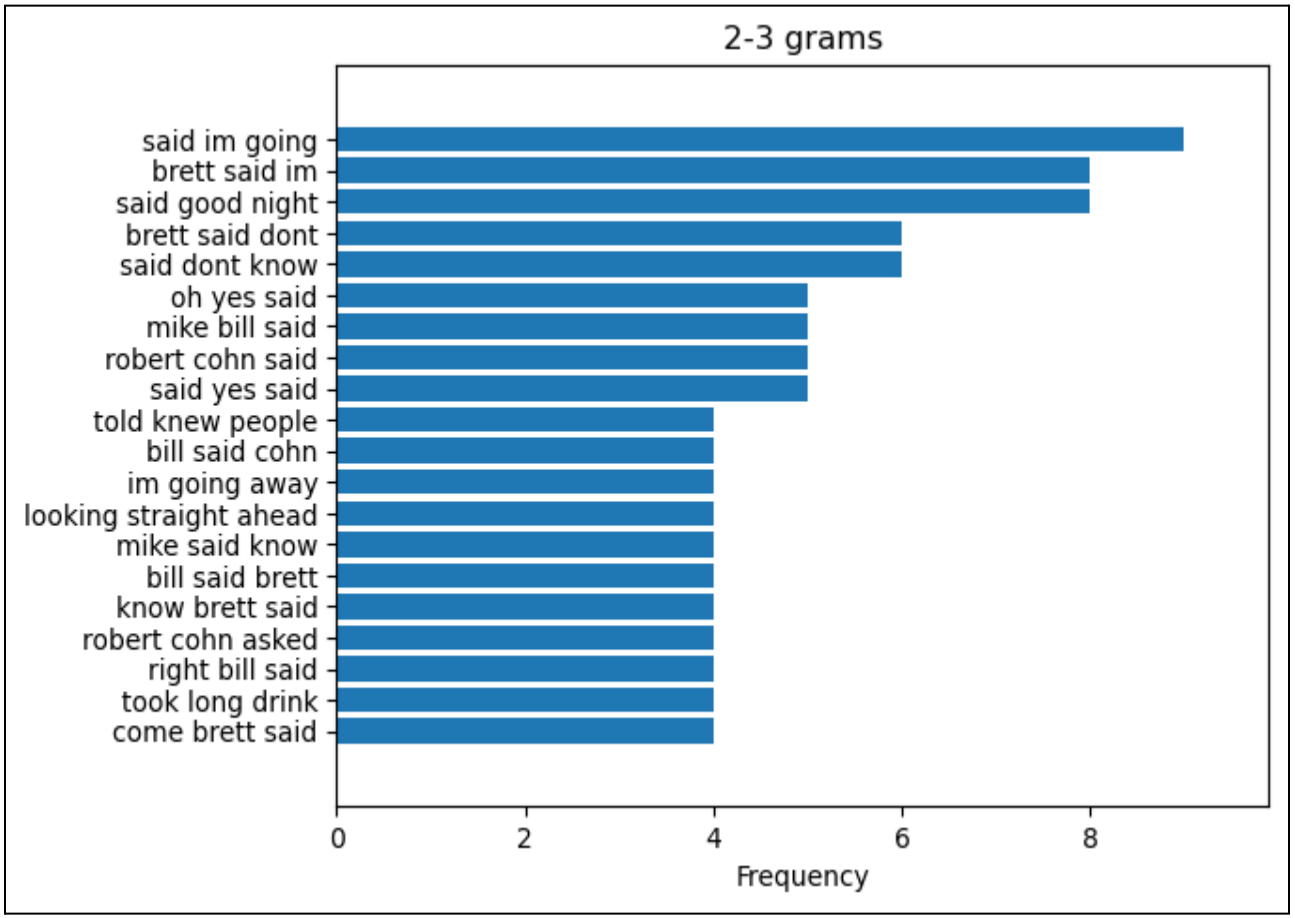
The bigrams and trigrams are displayed in the following figures and tables. As trigrams are less common than bigrams, their numbers drop off quickly, but the trend between these different analyses is clear. Ngrams with *said* dominate the landscape, taking the top three slots with the bigrams and the top nine slots with trigrams. This finding backs up the assessment from the word cloud that the novel is built off of dialogue. This approach supports Hemingway's "iceberg" method of hinting at the immensity of emotions and drama under the surface without explicitly detailing their sweeping range.

Figure 8 and Table 2: 20 most frequent bigrams



first	second	frequencies	ratio
bill	said	130	0.0045164
brett	said	122	0.00423847
mike	said	87	0.00302251
robert	cohn	55	0.00191078
dont	know	52	0.00180656
im	going	40	0.00138966
cohn	said	40	0.00138966
said	bill	39	0.00135492
said	brett	38	0.00132018
san	sebastian	33	0.00114647
said	im	32	0.00111173
said	dont	32	0.00111173
dont	think	28	0.00097276
yes	said	27	0.00093802
pedro	romero	23	0.00079906
said	mike	19	0.00066009
shook	hands	17	0.00059061
whats	matter	17	0.00059061
good	night	16	0.00055586
bill	asked	16	0.00055586

Figure 9 and Table 3: 20 most frequent trigrams



first	second	third	frequencies	ratio
said	im	going	9	0.00031268
brett	said	im	8	0.00027794
said	good	night	8	0.00027794
brett	said	dont	6	0.00020846
said	dont	know	6	0.00020846
oh	yes	said	5	0.00017371
mike	bill	said	5	0.00017371
robert	cohn	said	5	0.00017371
said	yes	said	5	0.00017371
told	knew	people	4	0.00013897
bill	said	cohn	4	0.00013897
im	going	away	4	0.00013897
looking	straight	ahead	4	0.00013897
mike	said	know	4	0.00013897
bill	said	brett	4	0.00013897
know	brett	said	4	0.00013897
robert	cohn	asked	4	0.00013897
right	bill	said	4	0.00013897
took	long	drink	4	0.00013897
come	brett	said	4	0.00013897

Word Embeddings:

Modern NLP packages, from organizations like spaCy and NLTK, incorporate semantic information into numerical representations of words. Text strings are tokenized into vectors. In spaCy, these vectors are lists of integers with 300 dimensions per word (See Figure 9). For this project, spaCy's medium core web model was used as that is the smallest one that has word embeddings: https://spacy.io/models/en. Words that are closer in meaning have more similar vectors.

Figure 9: The word *worry* and half of its vector from spaCy

```
worry [ 2.9654     2.7027    -2.4714    -1.0352    -0.079387  -0.72472
  0.53368    3.8428    -1.3603     2.3727     0.22586    0.75641
 -4.1174     0.83035    0.73932   -1.948      0.27187   -3.479
 -1.3285     1.1281    -0.53185    1.2922     0.53875   -3.2607
 -2.4816    -1.059     -0.81081   -2.788     -3.3112     4.8863
  0.13508   -5.8231     0.74262   -2.3775     3.5092     1.4314
  0.092381   0.64814    5.9549     4.9481    -1.8308     0.028828
  2.5295     1.3199    -2.2144    -0.91651    2.4754    -2.894
 -2.5452     0.48501    0.091526  -3.0365     2.2395    -3.6397
 -2.7671     1.0851     0.60659    3.4097     0.0090092  1.5626
  3.1725    -0.69717   -1.6433    -1.811     -1.0557    -1.0214
  0.58334    0.46305    2.307      1.2694     1.1162     2.6585
 -2.9381    -1.3663     1.7924    -2.3868     1.9842     0.44663
  3.5436     0.32388   -0.87528   -0.96563    2.767      2.3729
  2.0346    -0.8488     0.015217  -5.0707     0.21105    2.6214
 -1.3929     2.8451    -3.3657     1.1677    -0.441     -5.6514
 -0.59095    1.7608    -0.74169   -1.6772     0.52922    1.4826
  0.21566    1.1929    -0.3963     1.2581    -0.44234    1.3542
 -0.15589    0.40392   -2.3522    -1.302     -1.555     -4.6697
 -0.23593    4.8172    -5.3577    -3.7915     2.7631     0.63667
 -1.5845    -1.6256    -5.1559    -1.2487    -3.9772     3.1195
  1.1596     5.6142    -2.2377    -0.84486   -0.24609    1.9577
 -1.1008     1.7674     2.3359    -2.0569    -4.6275    -3.7086
  5.1245    -4.2225    -0.75321    2.0614     1.9893    -0.23126
 -4.3271    -2.7021    -1.9307    -1.1503    -1.0111    -0.2155   ]
```

When set up well, these vector strings can be contrasted with each other to find comparisons like similarities. As part of this analysis, I compared the first and last chapter of the novel to see if the overall vector average of each (and hence, their meaning) were similar to each other. By tokenizing each chapter, I then assessed their mean vectors with spaCy's similarity method to find that the similarity score is 0.96 (with 1 being identical). This finding indicates that the first and last chapter of the novel are semantically highly similar to each other.

Masking Sentences:

One of BERT's primary purposes is to guess words that have been covered up in a sentence, so-called Masked Language Modeling. Its other primary purpose is to predict the next sentence after being given some input. An advantage of this particular method is that BERT scans the sentence from left to right and from right to left; the *B* in the acronym stands for *bidirectional*. With this skill being a foundational part of its knowledge set, I put BERT to use to see if training it further on a Hemingway text would allow for better prediction than the untrained model. With this approach, we have the actual words Hemingway wrote as the truth value for each sentence. If either the trained or untrained model predicts the truth value, accuracy would

be 100% in that instance. For this project, I am using distilbert-base-uncased: https://huggingface.co/distilbert-base-uncased, as it has a smaller number of parameters than the larger BERT model. It only has 66 million parameters.

Actual Hemingway Sentence: `'bullfights good bullfighters stayed montoyas hotel'`

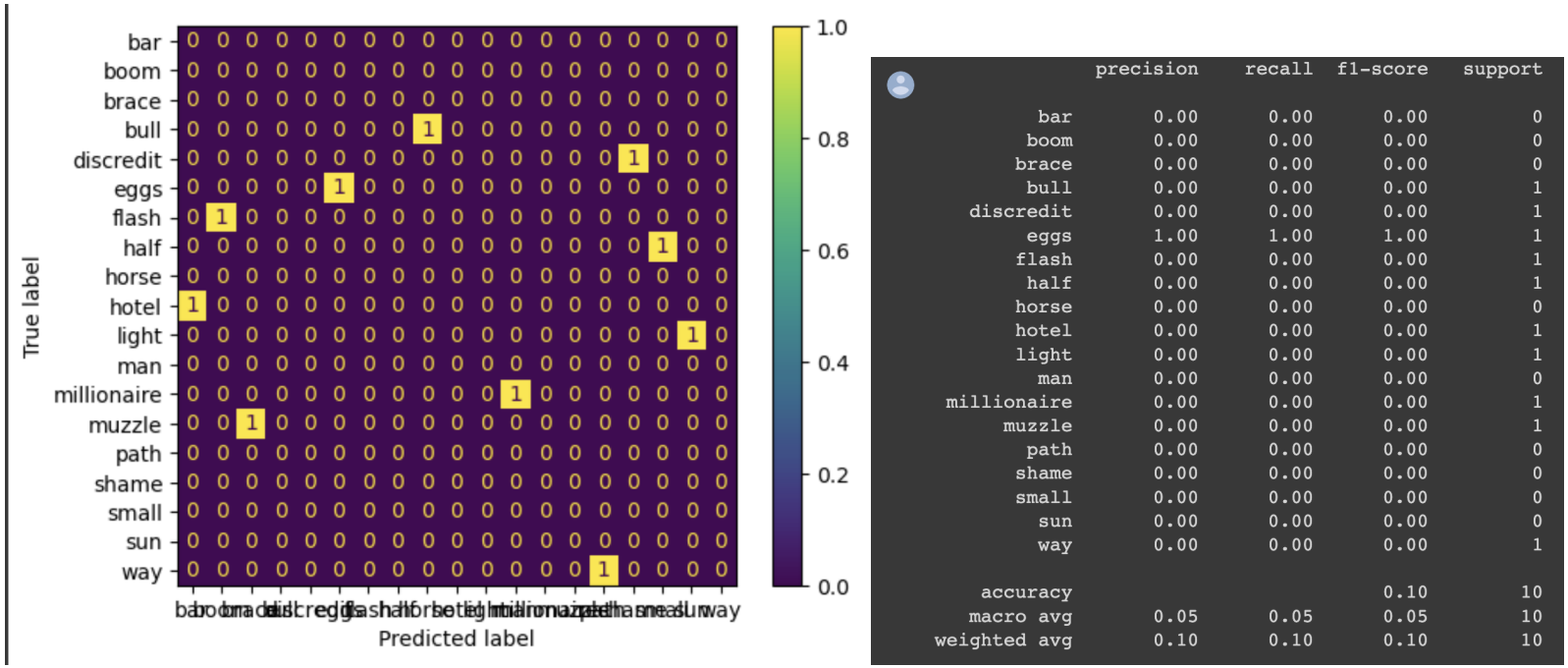
Masked Hemingway Sentence: `bullfights good bullfighters stayed montoyas [MASK]`

The exercise for this project was conducted with cleaned text (removing punctuation, stop words, and differences in case), but the DistilBERT model contains both punctuation and stop words, so in the end, the full first half of the novel was used to train the model, uncleaned.

Unmasking was performed on 10 Hemingway sentences from the second half of the novel in two different runs. In the first run, the out-of-the-box distilbert-base-uncased was used to predict the masked words. In the second run, the model was trained on the second half of *The Sun Also Rises* and then was used to predict the masked words from the same 10 sentences. The fine-tuned model can be found here: https://huggingface.co/khazen2/DistilBERT_Hemingway_SAR.

Although confusion matrices are commonly used to assess the quality of predictions, they work best when predictions are either binary or at least constrained to a few categories. Most traditional representations of a confusion matrix show simply four boxes, with true positives, false negatives, true negatives, and false positives. From such a confusion matrix, scores such as precision, recall, and F1 can be calculated with standard metrics from sklearn. With 10 predicted words and 10 true words, the confusion matrix is less useful, as the comparisons are based on similarity of the text string itself, and exact matches get a score of 1 and any mismatch gets a score of 0. If exact matches are the lowest threshold for acceptable predictions, this would be the approach to use for assessment between the models. See Figure 10 for an illustration of an overly crowded confusion matrix and report with such a high threshold of acceptable predictions.

Figure 10: A crowded confusion matrix and its table of predictions

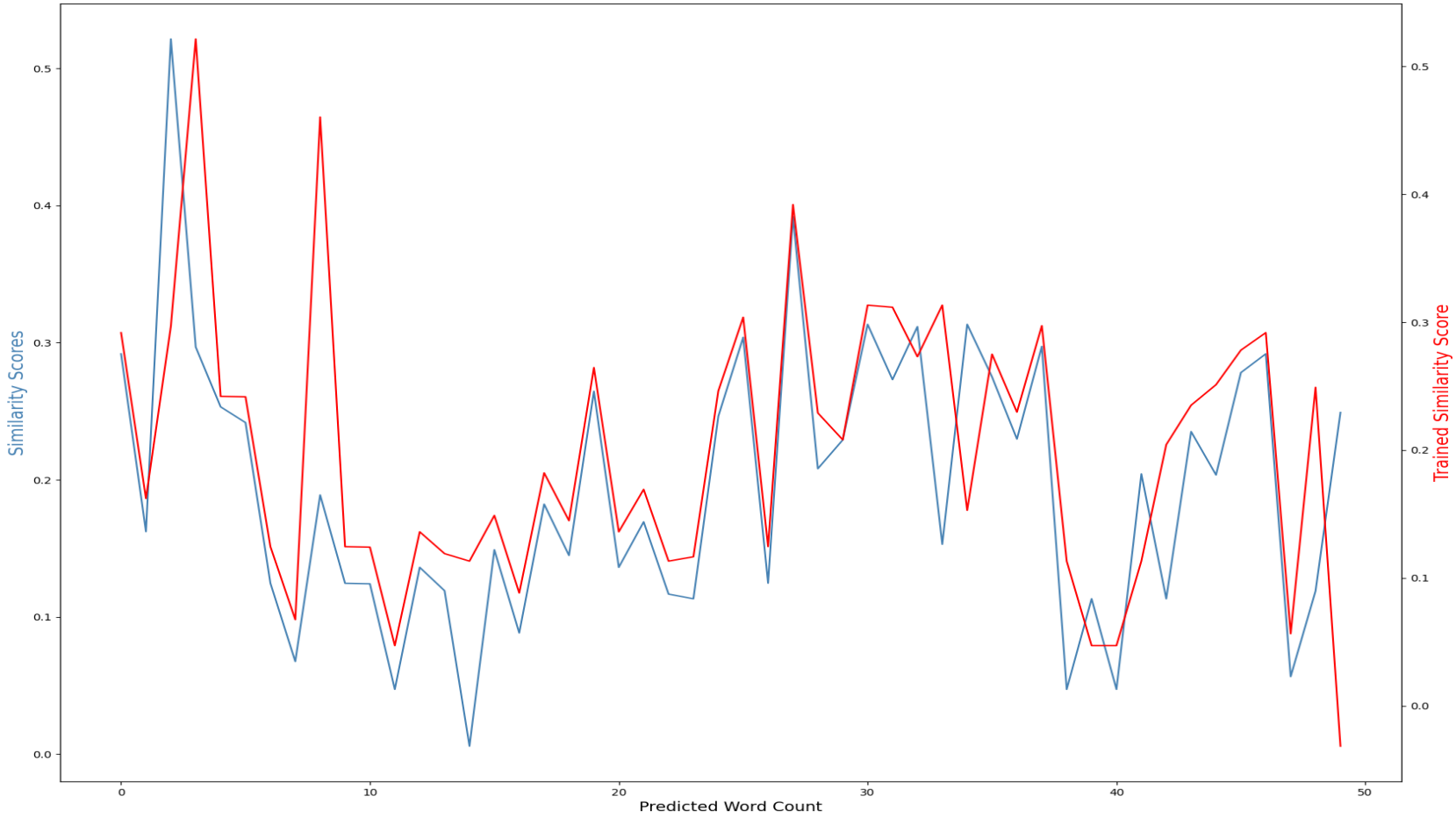


	precision	recall	f1-score	support
bar	0.00	0.00	0.00	0
boom	0.00	0.00	0.00	0
brace	0.00	0.00	0.00	0
bull	0.00	0.00	0.00	1
discredit	0.00	0.00	0.00	1
eggs	1.00	1.00	1.00	1
flash	0.00	0.00	0.00	1
half	0.00	0.00	0.00	1
horse	0.00	0.00	0.00	0
hotel	0.00	0.00	0.00	1
light	0.00	0.00	0.00	1
man	0.00	0.00	0.00	0
millionaire	0.00	0.00	0.00	1
muzzle	0.00	0.00	0.00	1
path	0.00	0.00	0.00	0
shame	0.00	0.00	0.00	0
small	0.00	0.00	0.00	0
sun	0.00	0.00	0.00	0
way	0.00	0.00	0.00	1
accuracy			0.10	10
macro avg	0.05	0.05	0.05	10
weighted avg	0.10	0.10	0.10	10

For a more semantically nuanced comparison of the accuracy of the predictions, the similarity score approach from spaCy was used to compare the predicted words and the true words. The similarity scores are based on the word's embedding. The full table of predictions can be found below. In the end, each true word was compared with five predicted words for 10 masked sentences, making a total of 50 predictions for the base and trained models.

The similarity scores from the untrained and trained models were compared with each other to see if they are significantly different using the Mann-Whitney test. This test found a U-statistic of 1245 and a p-value of 0.97. These findings indicate that the null hypothesis is resoundingly confirmed. There are a few areas where the direction of the difference is that the trained model did a better job predicting the masked word because it yielded better similarity scores, but these are few and far between. The overall result is that the similarity scores from the trained model are not much different from those of the base model. The scores themselves can be seen in Figure 11:

Figure 11: Untrained and Trained Similarity Scores



Conclusion:
The modern suite of NLP tools offer powerful options for prediction and generation. Statistical approaches and large datasets for training have allowed researchers to produce useful models, including the one applied in this project, BERT. With one of the 20th century's most important novels as a testing ground, this project assessed Ernest Hemingway's writing style and used a large-language model. The primary question was whether training the model on Hemingway's writing helped to improve the accuracy of the predictions.

Appendix:

Table 3: Similarity scores of the two models' predictions

Base DistilBERT			Fine-tuned Model	
Predicted Word	Similarity Scores	True Word	Similarity Scores	Predicted Word
carries	0.2918593414	put	0.2918593414	carries
collects	0.1621982666	put	0.1621982666	collects
pulls	0.5215437336	put	0.2968177014	handles
handles	0.2968177014	put	0.5215437336	pulls
opens	0.253190749	put	0.2420324967	steals
prepares	0.2417159155	brought	0.2417159155	prepares
drinks	0.124599894	brought	0.124599894	drinks
orders	0.06745186402	brought	0.06745186402	orders
delivers	0.1889055538	brought	0.4605056586	brings
serves	0.1245039204	brought	0.1245039204	serves
follows	0.1241230271	crossed	0.1241230271	follows
of	0.04713127512	crossed	0.04713127512	of
crosses	0.1361190863	crossed	0.1361190863	crosses
becomes	0.1190624598	crossed	0.1190624598	becomes
along	0.00571463249	crossed	0.1132015856	joins
won	0.1489174389	wore	0.1489174389	won

wins	0.08828834335	wore	0.08828834335	wins
received	0.1822173118	wore	0.1822173118	received
awarded	0.1448207246	wore	0.1448207246	awarded
earned	0.2645296407	wore	0.2645296407	earned
crosses	0.1361190863	rose	0.1361190863	crosses
enters	0.1692948248	rose	0.1692948248	enters
drains	0.1166018887	rose	0.1132015856	joins
joins	0.1132015856	rose	0.1166018887	drains
reaches	0.2463236769	rose	0.2463236769	reaches
gets	0.3039218745	took	0.3039218745	gets
drinks	0.124599894	took	0.124599894	drinks
takes	0.3920284186	took	0.3920284186	takes
buys	0.2080964081	took	0.2290770901	wants
wants	0.2290770901	took	0.2080964081	buys
drenched	0.3133757281	watching	0.3133757281	drenched
fills	0.2730777905	watching	0.3117152345	filled
filled	0.3117152345	watching	0.2730777905	fills
flooded	0.1529512955	watching	0.3133757281	soaked
soaked	0.3133757281	watching	0.1529512955	flooded
drives	0.2750188755	blew	0.2750188755	drives
strikes	0.2297417139	blew	0.2297417139	strikes
blows	0.2972960711	blew	0.2972960711	blows
of	0.04713127512	blew	0.1132015856	joins
joins	0.1132015856	blew	0.04713127512	of
of	0.04713127512	had	0.04713127512	of
calls	0.2042616897	had	0.1132015856	joins
joins	0.1132015856	had	0.2042616897	calls
called	0.2351370356	had	0.2351370356	called

escapes	0.2035278892	had	0.2511670024	told
wears	0.278173724	poured	0.278173724	wears
carries	0.2918593414	poured	0.2918593414	carries
owns	0.05646003632	poured	0.05646003632	owns
uses	0.1188752262	poured	0.2490915855	makes
makes	0.2490915855	poured	-0.03148168291	has