First Vectorized Representation

TF-IDF, Word2Vec, Doc2Vec, and ELMo Embeddings

Ritesh Kumar

2024SP_MS_DSP_453-DL_SEC61: Natural Language Processing

Module 3

Assignment A.1

Nethra Sambamoorthi and Sudha BG

May 29, 2024

Approach

The goal of this assignment is to identify terms within each document that are strong candidates for inclusion in a corpus-wide vocabulary. This vocabulary will be used to create a final vector representation for clustering and/or classification. The process involves selecting three or four key terms from each document based on their importance and prevalence, followed by evaluating these selections using quantitative metrics like TF-IDF scores.

Dataset: The assignment utilizes two datasets:

1. First Dataset: Reviews and author information for the top-5 books on the list "Best Books of the Twentieth Century" from www.goodreads.com.
2. Second Dataset: Reviews and author information for the top-5 books on the list "The Best of the Best Romance Novels of the Twentieth Century" from www.goodreads.com.

Selected Terms: The focus of the assignment is on these ten terms, because their frequent occurrence across the documents:

- Novel: Central to describing the book genre.
- Author: Important for contextual information
- Fiction: Common category for most book reviewed
- Story: Critical for summarizing the plot.
- Romance: Significant due to the romantic elements in most of the books
- Love: Frequent theme in most narratives.
- Write: Frequently used in context of book reviews.
- Book: frequently used because we are analyzing book reviews.
- Historical: Most books are set in a historical setting.

- Words: Often used in book reviews.

We also keep an eye on these rarely used words in the book reviews:

- Price: could be used in some book reviews.
- Appeal: could be used to describe the appeal to a certain demography.
- Simple: could be used to describe various facets of a book.
- Crisis: could be used in book reviews to describe the plots.

We began by computing the TF-IDF scores for the words in the ten book reviews to measure their importance within the corpus. Next, we generated Word2Vec, Doc2Vec, and ELMo embeddings to capture semantic relationships and contextual information. We then compared the clustering results of the ten most frequent words using these different embeddings. This comparison helped us understand how each embedding method represented the key terms and their relationships within the text.

TF-IDF: For computing the TF-IDF, we cleaned the text using the `clean_text` function. The function `clean_text` processed a given text by removing newline characters, apostrophes, and quotation marks, and replaced hyphens within words with a temporary token. It identified and temporarily replaced URLs with placeholders, removed all punctuation except hyphens, then restored the temporary hyphen token. The function also removed extra whitespaces, restored the URLs from their placeholders, and converted the text to lowercase.

After this cleaning process, the text was lemmatized, reducing words to their base or dictionary form, before creating the word corpus. Finally, the cleaned and lemmatized text was returned.

Next, we identified the two and three-term nouns and noun phrases and updated our word corpus. The total number of words in the word corpus turned out to be 1461. Finally, we computed the TF-IDF for the entire word corpus.

```
# Compute TF, IDF, and TF-IDF
tf_idf_dict = {}
for word, count in sorted_words_dict.items():
    tf = count / total_words
    idf = math.log(total_docs / (1 + doc_freq[word]))
    tf_idf_dict[word] = tf * idf
```

Ten words with the highest TF-IDF are:

```
# top ten of tf_idf_df
top_ten = tf_idf_df.nlargest(10, 'TF-IDF')
print(top_ten)
```

	Word	Word Count	TF-IDF
0	novel	42	0.021223
1	author	29	0.014654
2	fiction	28	0.014149
3	story	28	0.014149
4	romance	26	0.013138
5	love	25	0.012633
6	write	23	0.011622
7	book	22	0.011117
8	historical	21	0.010612
9	word	21	0.010612

And, the first 20 words with the lowest TF-IDF and word-count are:

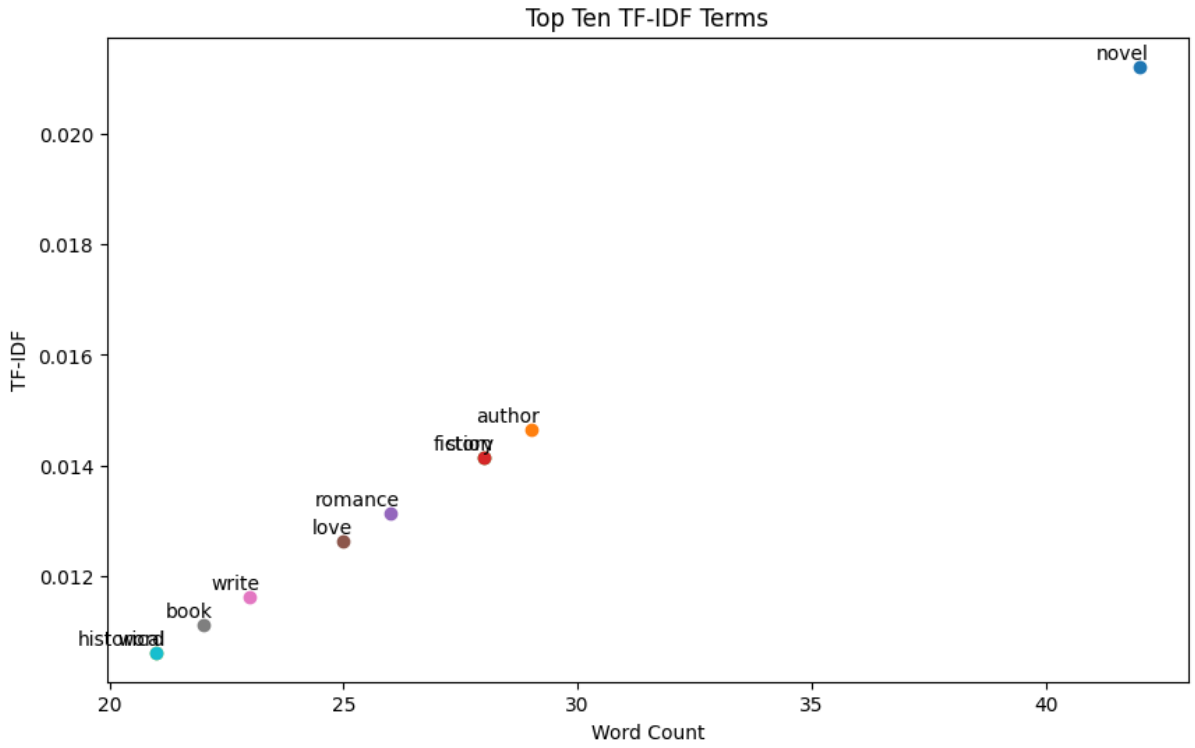
```
# last ten of tf_idf_df
last_20 = tf_idf_df.nsmallest(20, 'TF-IDF')
print(last_20)
```

	Word	Word Count	TF-IDF
584	538	1	0.000505
585	price	1	0.000505
586	unforgettable	1	0.000505
587	sleepy	1	0.000505
588	crisis	1	0.000505
589	conscience	1	0.000505
590	rock	1	0.000505
591	instant	1	0.000505
592	academy	1	0.000505
593	dramatic	1	0.000505
594	root	1	0.000505
595	innocence	1	0.000505
596	cruelty	1	0.000505
597	pathos	1	0.000505
598	18	1	0.000505
599	regional	1	0.000505
600	claim	1	0.000505
601	universal	1	0.000505
602	appeal	1	0.000505
603	simple	1	0.000505

As expected, words with the highest TF-IDF and word-count are: 'novel', 'author', 'fiction', 'story', 'romance', 'love', 'write', 'book', 'historical', 'words'.

And, as expected, words with the lowest TF-IDF and word-count are: ‘price’, ‘appeal’, ‘simple’, ‘crisis’.

Plotting the words with the highest TF-IDF and word-count:



The plot shows the top ten terms based on their TF-IDF values, with "novel" having the highest TF-IDF score, indicating its strong relevance despite a moderate word count. This suggests that "novel" is a key term in the corpus. Other terms like "author," "fiction," "romance," and "history" also appear, each with varying word counts and TF-IDF values. Terms such as "author" and "fiction" have high TF-IDF scores, signifying their importance in the text. The distribution reveals a positive correlation between word count and TF-IDF, suggesting that more frequently used terms tend to have higher importance.

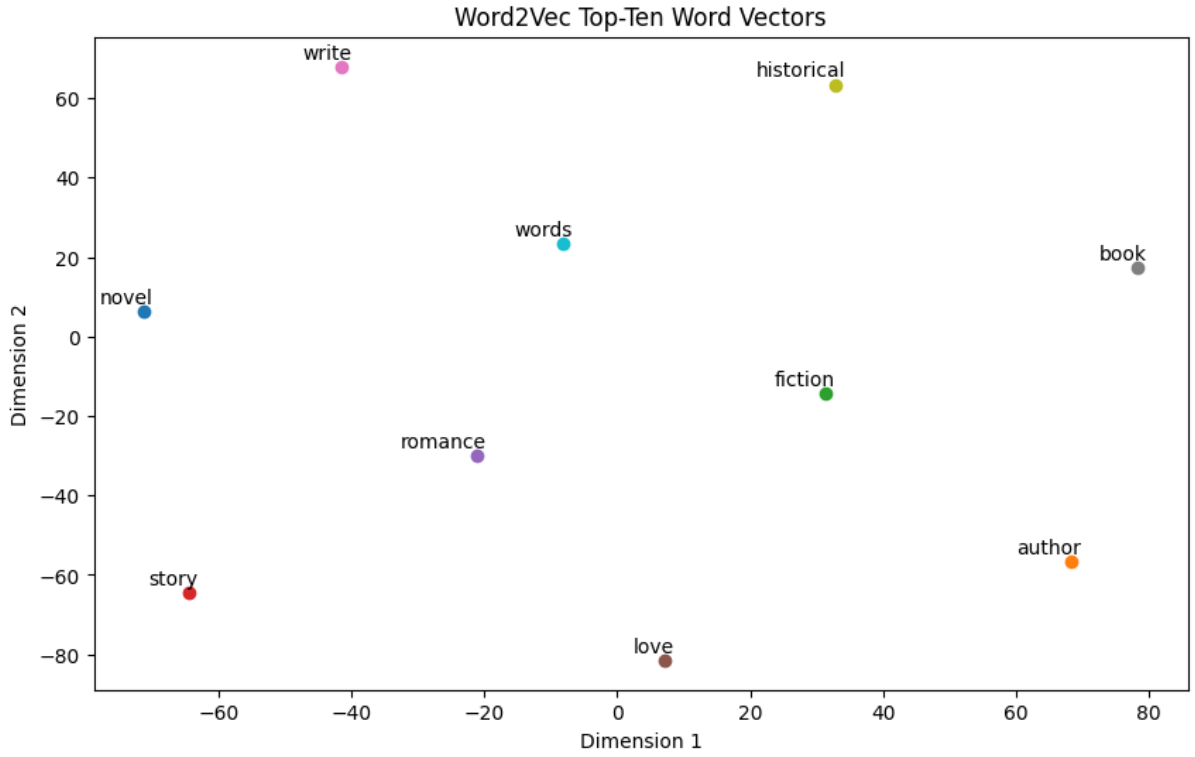
Word2Vec:

The function `clean_text` processed a given text by removing newline characters and apostrophes, replacing hyphens with blank spaces, and removing quotation marks. It stripped the text of leading and trailing whitespaces. Initially, it removed newline characters and replaced them with spaces. Apostrophes were eliminated, and hyphens were replaced with spaces, followed by the removal of any remaining hyphens with spaces. Quotation marks were also removed. Finally, the function ensured that extra whitespaces at the beginning and end of the text were stripped. The cleaned text was then returned.

However, stop words were not removed, nor were the words lemmatized. This was because retaining stop words and original word forms preserves the full context and syntactical structure of the text, which is crucial for models like Word2Vec, Doc2Vec, and ELMo that rely on context for generating meaningful embeddings. Stop words help maintain grammatical relationships, while original word forms convey nuances lost through lemmatization. This approach ensured better performance in downstream tasks like sentiment analysis and document classification, providing richer, more interpretable embeddings. Additionally, it allowed for consistent evaluation across different embedding methods, enhancing comparability and overall understanding.

The function trained a Word2Vec model using a corpus of tokenized sentences, specifying parameters such as vector size of 100, window size of 5, minimum word count of 1, and 4 worker threads. The trained model was then saved to a file named "word2vec.model." It extracted vectors for the top ten TF-IDF words that were present in the model's vocabulary. The extracted word vectors were converted into a NumPy array. To reduce the dimensionality of these vectors to 2D, t-SNE was employed with a perplexity value ensuring it was less than the number of valid words. Finally, the function plotted the 2D vectors using matplotlib, labeling each point with the corresponding word. This visualization displayed the spatial

relationships between the top ten TF-IDF words in the reduced 2D space, highlighting their semantic similarities and differences.

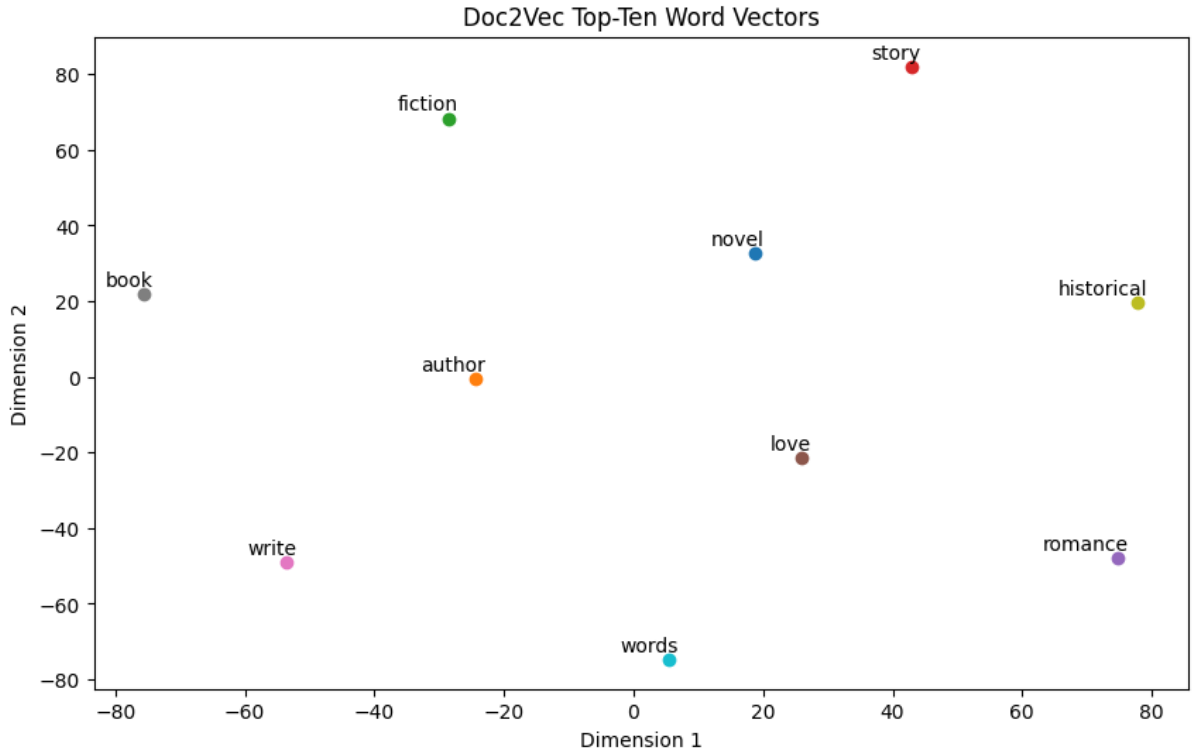


The plot visualizes the 2D projections of the top ten TF-IDF words using Word2Vec embeddings reduced by t-SNE. Each point represents a word, with their positions reflecting semantic relationships. "Author" and "book" are closely located, suggesting a semantic similarity. "Story" and "love" are situated near each other, indicating a thematic connection. Words like "historical" and "write" are more isolated, implying distinct contexts. The spread of terms like "novel," "romance," and "fiction" demonstrates their related but distinct meanings. This visualization helps in understanding the semantic space, highlighting how these terms are contextually related based on their usage in the corpus.

Doc2Vec:

After cleaning the data in the same manner as for word2Vec, the doc2vec function tagged each document in a dataset by splitting the text into words and assigning a unique tag to each

document using the `TaggedDocument` class. It then trained a Doc2Vec model with parameters including a vector size of 100, window size of 5, minimum word count of 1, 4 worker threads, and 20 epochs. The trained model was saved to a file named "doc2vec.model." The function extracted vectors for the top ten TF-IDF words that were present in the model's vocabulary, printing a message if a word was not found. These word vectors were converted into a NumPy array. Using t-SNE, the dimensionality of these vectors was reduced to 2D, with a perplexity value ensuring it was less than the number of valid words. Finally, the function plotted the 2D vectors using matplotlib, labeling each point with the corresponding word, visualizing their spatial relationships and semantic similarities.

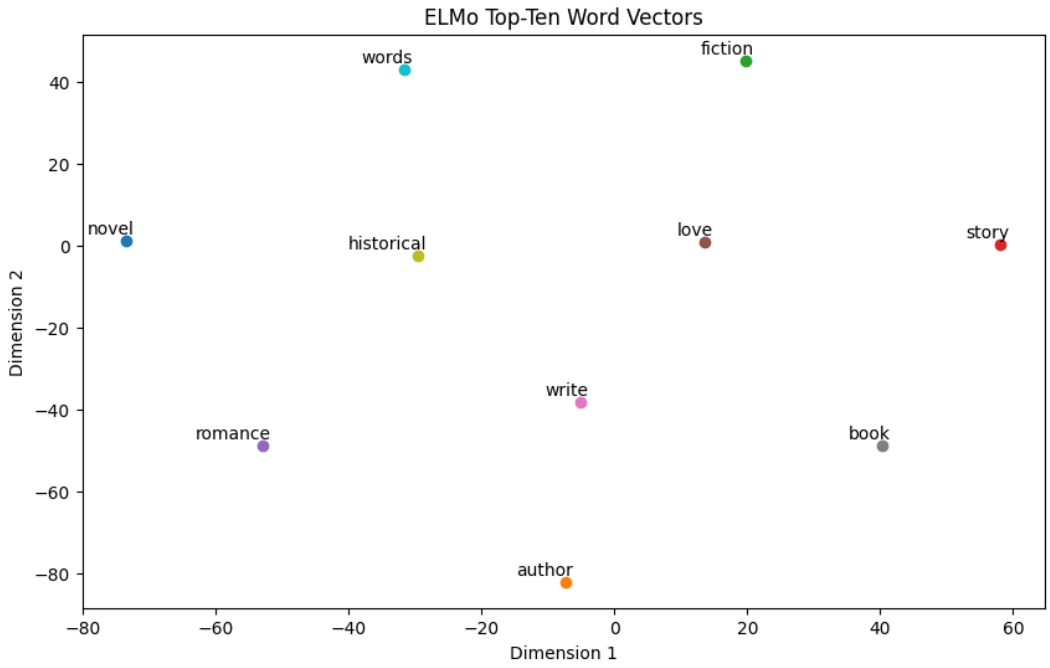


The plot visualizes the 2D projections of the top ten TF-IDF words using Doc2Vec embeddings reduced by t-SNE. Each point represents a word, with their positions indicating semantic relationships. "Author" and "book" are relatively close, suggesting a contextual similarity. "Story" and "fiction" are also near each other, highlighting a thematic connection. Words like "historical" and "romance" are more isolated, implying distinct contexts. The

spread of terms such as "novel," "write," and "love" demonstrates their related but distinct meanings. This visualization helps in understanding the semantic space, showing how these terms are contextually related based on their usage in the corpus.

ELMo:

After cleaning the data in the same manner as for word2Vec, the ELMo function loaded the ELMo model from TensorFlow Hub to compute word embeddings. It defined a function `elmo_embedding` to generate ELMo embeddings for given text. The sentences from `df.sentences` were compiled into a comprehensive list. Using `TfidfVectorizer`, the function vectorized the text, limiting the features to 2500 and extracted the vocabulary. For each word in the vocabulary, the function computed ELMo embeddings and stored them. The function then filtered these word vectors to include only the top ten TF-IDF words. These filtered word vectors were converted to a NumPy array. Using t-SNE, the dimensionality of these vectors was reduced to 2D, with a perplexity value ensuring it was less than the number of valid words. Finally, it plotted the 2D vectors using matplotlib, labeling each point with the corresponding word to visualize their semantic relationships.



The plot visualizes the 2D projections of the top ten TF-IDF words using ELMo embeddings reduced by t-SNE. Each point represents a word, with their positions reflecting semantic relationships. "Author" and "write" are relatively close, suggesting a contextual similarity. "Story" and "love" are also near each other, indicating a thematic connection. Words like "words" and "fiction" are more isolated, implying distinct contexts. The spread of terms such as "novel," "romance," and "historical" demonstrates their related but distinct meanings. This visualization helps in understanding the semantic space, showing how these terms are contextually related based on their usage in the corpus. The arrangement of words reveals their semantic similarities and differences, providing insights into the relationships between these key terms.

Comparison:

Detailed Comparison of Word Placement Across the Four Plots:

1. Novel
    - TF-IDF: Highest TF-IDF score, indicating it's the most significant term.
    - Word2Vec: Positioned at the bottom-left, suggesting it's semantically distinct from other terms in this embedding space.
    - Doc2Vec: Located in the center, indicating a balanced contextual relationship with other terms.
    - ELMo: Found at the top-left, showing a unique contextual meaning within the ELMo embeddings.
2. Author
    - TF-IDF: Middle-range TF-IDF score, indicating moderate significance.
    - Word2Vec: Placed at the bottom-right, far from "write" and "book," indicating a distinct context.

- Doc2Vec: Positioned at the bottom-center, closer to "write" and "book," suggesting a stronger contextual link.
- ELMo: Located at the bottom-center, showing a similar context to Doc2Vec, but more nuanced.

3. Fiction
    - TF-IDF: Middle-range TF-IDF score.
    - Word2Vec: Center-right, indicating some semantic similarity with terms like "book."
    - Doc2Vec: Top-center, showing a strong relationship with "story" and "novel."
    - ELMo: Top-center, suggesting a contextual similarity with "story" and "novel."
4. Story
    - TF-IDF: Middle-range TF-IDF score.
    - Word2Vec: Bottom-center, reflecting a general narrative context.
    - Doc2Vec: Top-right, indicating a close relationship with "fiction" and "novel."
    - ELMo: Top-right, maintaining a similar narrative context as in Doc2Vec.
5. Romance
    - TF-IDF: Middle-range TF-IDF score.
    - Word2Vec: Mid-left, showing a semantic connection to terms like "love."
    - Doc2Vec: Bottom-right, placed near "love," indicating a strong contextual relationship.
    - ELMo: Bottom-left, showing a nuanced contextual relationship with "love."
6. Love
    - TF-IDF: Middle-range TF-IDF score.
    - Word2Vec: Bottom-right, near "romance," indicating a semantic similarity.
    - Doc2Vec: Bottom-center, clustered with "romance," showing a close contextual relationship.

- ELMo: Bottom-center, maintaining proximity to "romance" and indicating a similar context.

7. Historical
    - TF-IDF: Lower-range TF-IDF score.
    - Word2Vec: Top-right, suggesting a distinct historical context.
    - Doc2Vec: Center-right, showing some semantic distance from "story" and "fiction."
    - ELMo: Top-right, reflecting a historical context, somewhat distinct from other terms.
8. Book
    - TF-IDF: Lower-range TF-IDF score.
    - Word2Vec: Top-right, indicating a relationship with "author" and "write."
    - Doc2Vec: Top-left, suggesting a broader contextual meaning.
    - ELMo: Top-right, similar to Word2Vec, showing a link to "author" and "write."
9. Write
    - TF-IDF: Lower-range TF-IDF score.
    - Word2Vec: Top-left, indicating a distinct context related to writing activities.
    - Doc2Vec: Bottom-left, showing a close relationship with "author."
    - ELMo: Bottom-center, reflecting a contextual meaning related to "author" and "book."
10. Words
    - TF-IDF: Lower-range TF-IDF score.
    - Word2Vec: Center, suggesting a general linguistic context.
    - Doc2Vec: Bottom-center, indicating a broad relationship with other terms.
    - ELMo: Top-center, reflecting a contextualized meaning within the corpus.

Conclusion:

Novel is consistently shown as highly significant but varies in contextual relationships across different methods, indicating different nuanced meanings.

Author shows distinct placements, with Word2Vec suggesting a more isolated context compared to the more connected positioning in Doc2Vec and ELMo.

Fiction and story are closely related in Doc2Vec and ELMo, reflecting a narrative context.

Romance and love are consistently placed near each other in Word2Vec, Doc2Vec, and ELMo, indicating strong semantic and contextual relationships.

Historical and book show varying distances, with ELMo and Word2Vec highlighting distinct contexts and Doc2Vec showing broader relationships.

Write and words exhibit different relationships across embeddings, with "write" being more contextually connected to "author" and "book" in Doc2Vec and ELMo.

These comparisons demonstrate that each technique captures different aspects of word meanings and relationships, providing a comprehensive understanding of term significance, local semantic relationships, and contextual nuances within the corpus. By analyzing the TF-IDF chart, we can identify which terms are the most important based on their frequency and relevance in the text, giving us a clear picture of the key themes and concepts. The Word2Vec embeddings offer insights into the semantic similarities between words, revealing how terms are related based on their usage in similar contexts, which helps in understanding the underlying connections and associations between different concepts.

The Doc2Vec embeddings extend this understanding by incorporating document-level context, allowing us to see how words relate to the overall themes and ideas expressed in entire documents, rather than just individual word usage. This provides a richer and more

holistic view of the text's structure and the relationships between terms. Finally, the ELMo embeddings capture the nuanced and dynamic meanings of words in different contexts, reflecting how the same word can have varying implications depending on its surrounding text. This contextualized representation offers the most detailed and intricate understanding of word relationships, showing how meanings shift and adapt in different scenarios.

Together, these techniques enable a multi-faceted analysis of the corpus, combining the strengths of each method to reveal both high-level themes and fine-grained details. This comprehensive approach ensures that we can appreciate the full complexity of the text, from the importance and frequency of individual terms to their semantic relationships and contextual nuances, providing a robust framework for text analysis and interpretation.