Exploring Trends and Sentiments in Historical Newspaper Articles

Dmitrii Krotov

2024-06-12

Contents

Introduction **2**

- **Dataset Overview** . . . 2
- **Scope of Analysis** . . . 2
- **Project Goals** . . . 2

1 Methods and Analysis **3**

- 1.1 **Data Cleaning and Preprocessing** . . . 3
- 1.2 **Data Exploration and Visualization** . . . 6
- 1.3 **Modeling Approach** . . . 9

2 Results **21**

- 2.1 **Temporal Distribution** . . . 21
- 2.2 **Sentiment Analysis** . . . 21
- 2.3 **Topic Modeling** . . . 22
- 2.4 **Exploring Relationships Between Words Using N-grams** . . . 22
- 2.5 **Word Embeddings** . . . 22

3 Conclusion **22**

- 3.1 **Summary of Findings** . . . 22
- 3.2 **Potential Impact** . . . 23
- 3.3 **Limitations** . . . 23
- 3.4 **Future Work** . . . 23

References **23**

Introduction

Dataset Overview

The Old Newspapers dataset is a cleaned subset of the HC Corpora newspapers. It contains over 16 million sentences/paragraphs in 67 languages from various newspapers. Each entry includes the following columns:

- Language: The language of the text.
- Source: The newspaper from which the text is extracted.
- Date: The date of the article containing the text.
- Text: The actual sentence or paragraph from the newspaper.

This dataset, available at https://www.kaggle.com/datasets/alvations/old-newspapers, provides a rich source of historical information that can be analyzed to uncover trends, sentiments, and relationships between words over time.

Scope of Analysis

This analysis focuses on English-language articles from the Old Newspapers dataset. The primary goal is to understand the temporal distribution of articles, perform sentiment analysis using different lexicons, explore the main topics through topic modeling, and investigate the relationships between words using n-grams.

Project Goals

Trend Analysis

We will analyze the temporal distribution of articles to identify any significant trends or patterns over time. This includes visualizing the number of articles published over the years and examining any notable spikes or drops.

Sentiment Analysis

We will perform sentiment analysis using the `syuzhet` package to gauge the emotional tone of the articles. Additionally, we will compare the results using different lexicons (`AFINN`, `bing`, and `nrc`) to identify common positive and negative words and understand the emotional trends over time.

Exploring Relationships Between Words

Using n-grams, we will explore the relationships between words in the articles. This involves tokenizing the text into bigrams, filtering out stop words, and visualizing the relationships between commonly co-occurring word pairs.

Word Embeddings

Map words to vector representations and explore semantic relationships using Word2Vec and t-SNE visualization.

1 Methods and Analysis

We will use the following libraries to assist with data manipulation, visualization, and model building:

```
# Load necessary libraries and install if missing
libraries <- c("httr", "jsonlite", "tidyverse", "data.table", "tm", "tidytext",
               "ggplot2", "lubridate", "topicmodels", "slam", "widyr",
               "ggraph", "igraph", "scales", "syuzhet", "text2vec", "Rtsne")
installed_libs <- libraries %in% rownames(installed.packages())

if (any(!installed_libs)) {
  install.packages(libraries[!installed_libs])
}

lapply(libraries, library, character.only = TRUE)
data(stop_words)
```

1.1 Data Cleaning and Preprocessing

1.1.1 Filtering English Articles

First, we filter the dataset to include only articles written in English.

```
# Filter for English articles
english_articles <- old_newspapers %>% filter(Language == "English")

# Save the filtered dataset for future use
write.csv(english_articles, file.path(intermediate_data_dl_path,
                                      "english_old_newspapers.csv"),
          row.names = FALSE)

# Load the filtered dataset into R
filtered_data_path <- file.path(intermediate_data_dl_path,
                                "english_old_newspapers.csv")
english_articles <- fread(filtered_data_path)
```

1.1.2 Handling Missing Data

Next, we check for and handle missing data, especially in the Text, Date, and Source fields.

```r
# Check for missing values in the Text, Date, and Source fields
missing_summary <- english_articles %>%
  summarise(
    total_rows = n(),
    missing_text = sum(is.na(Text) | Text == ""),
    missing_date = sum(is.na(Date) | Date == ""),
    missing_source = sum(is.na(Source) | Source == "")
  )

print(missing_summary)
```

```
##   total_rows missing_text missing_date missing_source
## 1    1010242            0            0              0
```

```r
# Remove rows with missing Text, Date, or Source values
english_articles_clean <- english_articles %>%
  filter(!is.na(Text) & Text != "" &
           !is.na(Date) & Date != "" &
           !is.na(Source) & Source != "")

# Save the cleaned dataset for future use
write.csv(english_articles_clean,
      file.path(intermediate_data_dl_path, "english_old_newspapers_clean.csv"),
          row.names = FALSE)

# Check the dimensions of the cleaned dataset
dim(english_articles_clean)
```

```
## [1] 1010242       4
```

```r
as_tibble(english_articles_clean)
```

```
## # A tibble: 1,010,242 x 4
##    Language Source             Date       Text
##    <chr>    <chr>              <chr>      <chr>
##  1 English  latimes.com        2012/04/29 "He wasn't home alone, apparently."
##  2 English  stltoday.com       2011/07/10 "The St. Louis plant had to close. It~
##  3 English  freep.com          2012/05/07 "WSU's plans quickly became a hot top~
##  4 English  nj.com             2011/02/05 "The Alaimo Group of Mount Holly was ~
##  5 English  sacbee.com         2011/10/02 "And when it's often difficult to pre~
##  6 English  cleveland.com      2012/04/27 "There was a certain amount of scoffi~
##  7 English  freep.com          2012/05/03 "14915 Charlevoix, Detroit"
##  8 English  nj.com             2011/02/02 "\"\"\"\"\"\"\"\"It's just another in a l~
##  9 English  chicagotribune.com 2012/01/05 "But time and again in the report, Su~
## 10 English  indystar.com       2012/05/04 "I was just trying to hit it hard som~
## # i 1,010,232 more rows
```

1.1.3 Text Normalization

We then normalize the text data by converting it to lowercase, removing punctuation, numbers, and excess whitespace.

```
normalize_text <- function(text) {
  text %>%
    tolower() %>%                              # Convert to lowercase
    str_replace_all("[[:punct:]]", " ") %>%   # Remove punctuation
    str_replace_all("[0-9]+", " ") %>%        # Remove numbers
    str_squish()                               # Strip excess whitespace
}

# Apply normalization to the Text column
english_articles_clean <- english_articles_clean %>%
  mutate(Text = map_chr(Text, normalize_text))

# Preview the normalized text
head(english_articles_clean$Text)
```

1.1.4 Tokenization and Removal of Stop Words

Finally, we tokenize the text into individual words and remove common stop words to focus on more meaningful words.

```
# Tokenization and stop words removal
tokenized_articles <- english_articles_clean %>%
  unnest_tokens(word, Text) %>%   # Tokenize the text into words
  anti_join(stop_words)   # Remove stop words

# Preview the tokenized data
head(tokenized_articles)
```

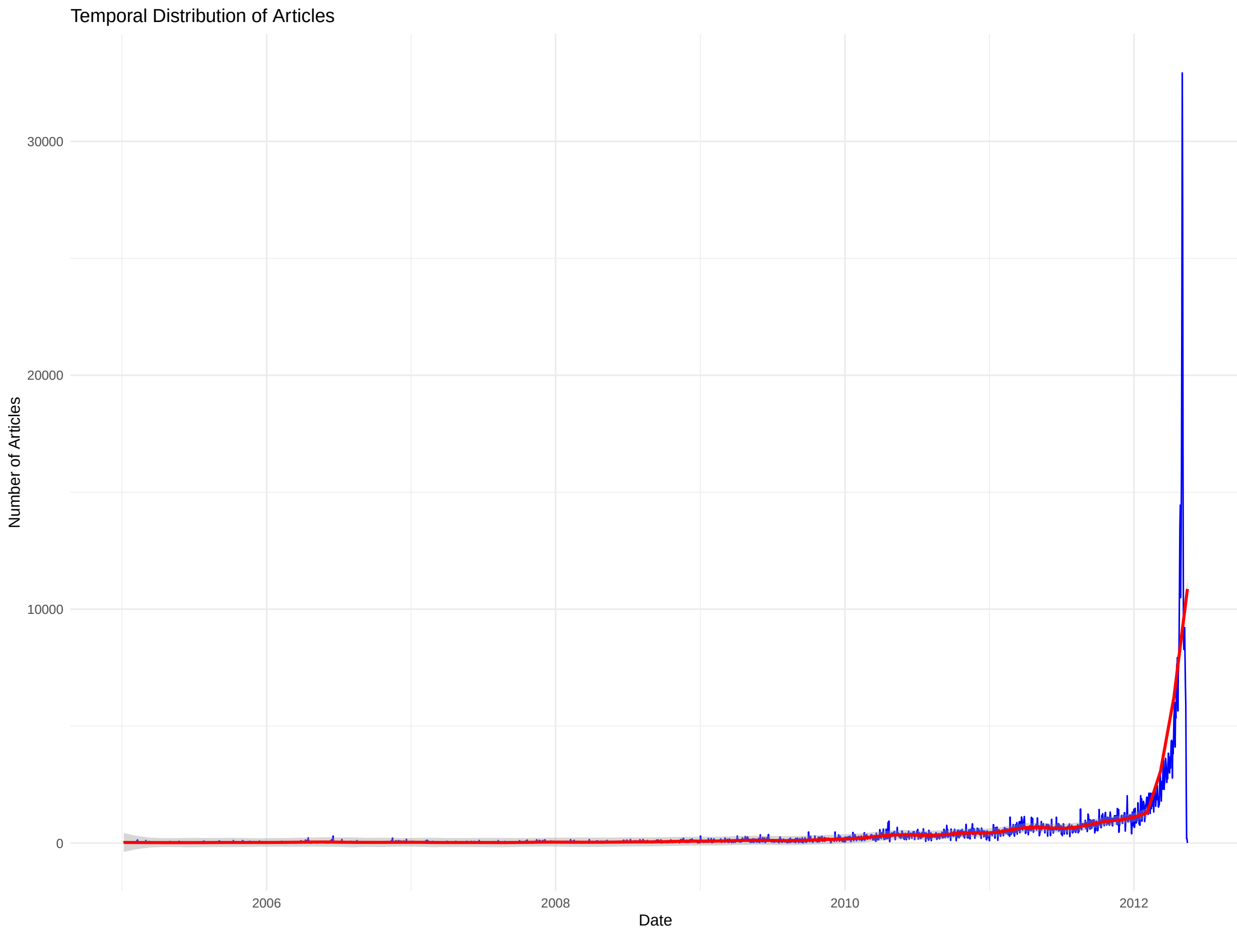
```
##    Language       Source       Date       word
##      <char>       <char>     <char>     <char>
## 1:  English  latimes.com 2012/04/29       wasn
## 2:  English  latimes.com 2012/04/29       home
## 3:  English  latimes.com 2012/04/29 apparently
## 4:  English stltoday.com 2011/07/10         st
## 5:  English stltoday.com 2011/07/10      louis
## 6:  English stltoday.com 2011/07/10      plant
```

1.2 Data Exploration and Visualization

1.2.1 Temporal Distribution

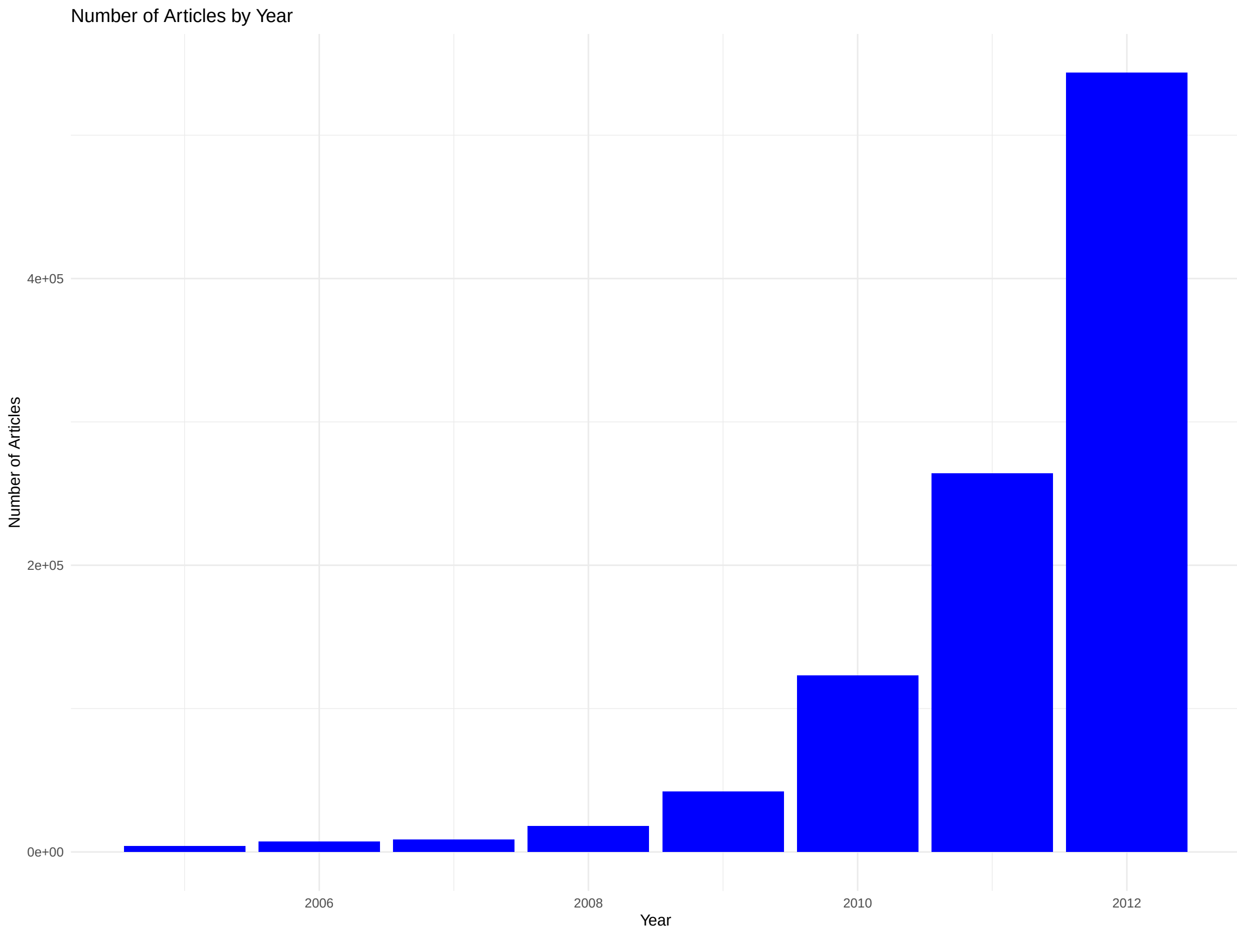
We analyze the distribution of articles over time to identify any significant trends or data collection biases.

```r
# Convert Date to Date type and add Year column
english_articles_clean <- english_articles_clean %>%
  mutate(Date = as.Date(Date, format = "%Y/%m/%d")) %>%
  mutate(Year = year(Date))

# Plot the distribution of articles over time with smoothing
temporal_distribution <- english_articles_clean %>%
  count(Date) %>%
  ggplot(aes(x = Date, y = n)) +
  geom_line(color = "blue") +
  geom_smooth(method = "loess", span = 0.1, color = "red") +
  labs(title = "Temporal Distribution of Articles",
       x = "Date",
       y = "Number of Articles") +
  theme_minimal(base_family = "sans", base_size = 11)
temporal_distribution
```



We also analyze the number of articles by year.

```
articles_by_year <- english_articles_clean %>%
  count(Year) %>%
  ggplot(aes(x = Year, y = n)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Number of Articles by Year",
       x = "Year",
       y = "Number of Articles") +
  theme_minimal(base_family = "sans", base_size = 11)
articles_by_year
```

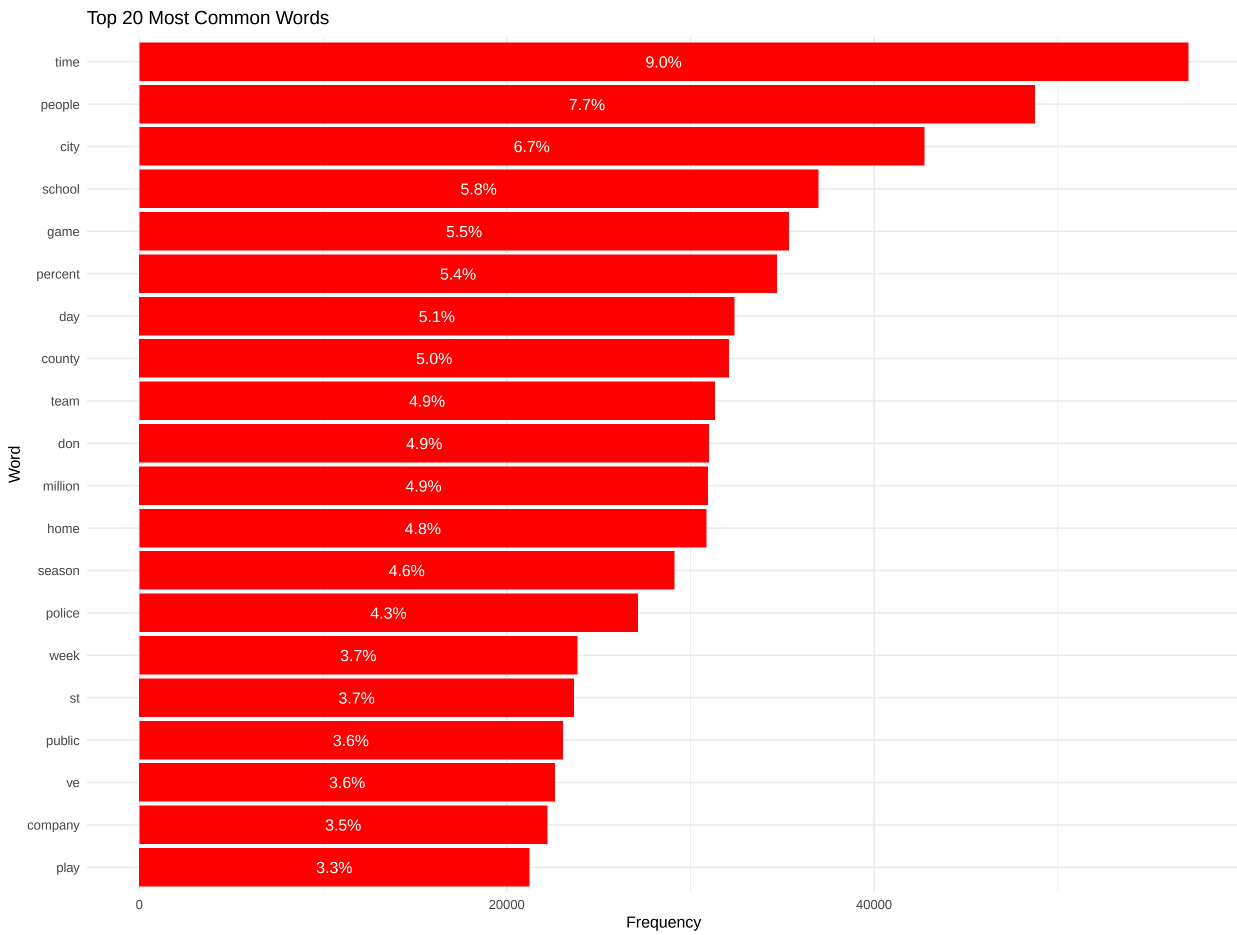


1.2.2 Initial Word Frequency Analysis

We identify the most common words and their frequencies to get an initial sense of prevalent topics or concerns.

```
# Calculate word frequency and proportion
word_frequency <- tokenized_articles %>%
  count(word, sort = TRUE) %>%
  top_n(20, n) %>%
  mutate(proportion = n / sum(n))

# Plot the most common words with proportions
word_frequency_plot <- word_frequency %>%
  ggplot(aes(x = reorder(word, n), y = n, label = scales::percent(proportion,
                                                         accuracy = 0.1))) +
  geom_bar(stat = "identity", fill = "red") +
  geom_text(aes(label = scales::percent(proportion, accuracy = 0.1)),
            position = position_stack(vjust = 0.5),
            color = "white",
            size = 4) +
```

```
  coord_flip() +
  labs(title = "Top 20 Most Common Words",
       x = "Word",
       y = "Frequency") +
  theme_minimal(base_family = "sans", base_size = 11)
word_frequency_plot
```



1.3 Modeling Approach

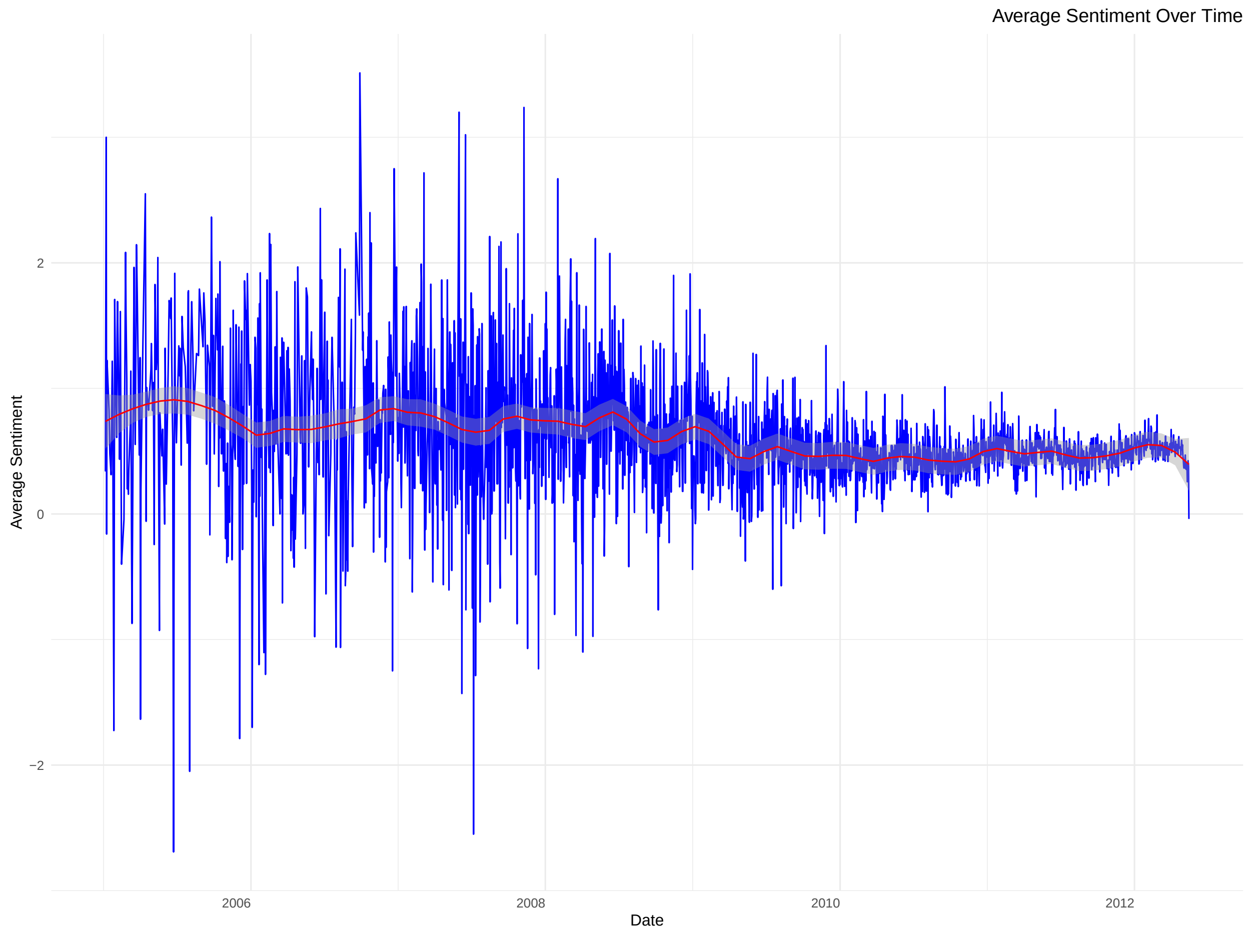
1.3.1 Sentiment Analysis

We perform sentiment analysis to gauge the emotional tone of articles over time using the `syuzhet` package, followed by a comparison using different sentiment lexicons (`AFINN`, `bing`, and `nrc`).These lexicons contain many English words, each assigned scores or categories indicating their sentiment or emotional tone:

- **nrc lexicon**: Categorizes words in a binary fashion ("yes"/"no") into categories such as positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

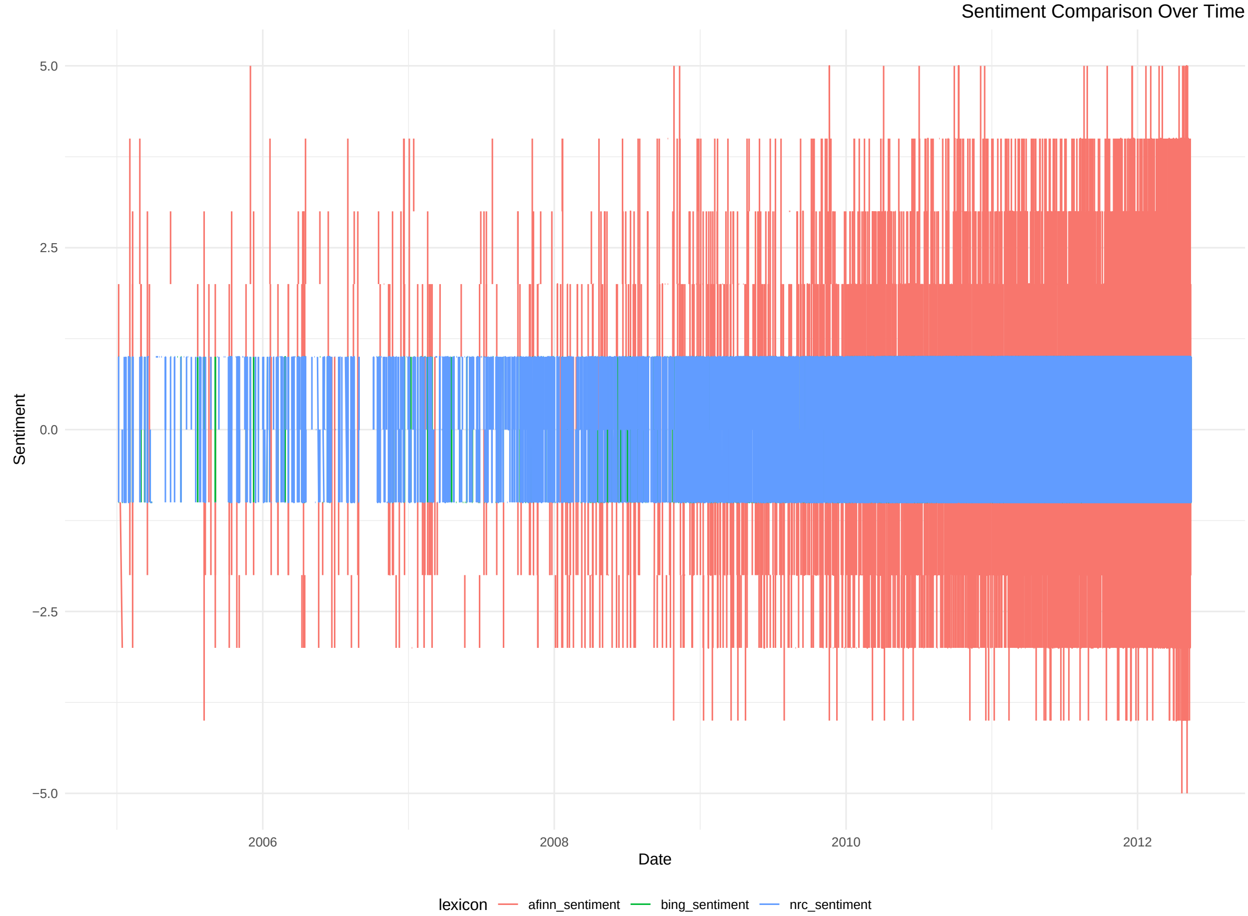
- **bing lexicon**: Categorizes words in a binary fashion into positive and negative categories.
- **AFINN lexicon**: Assigns words a score ranging from -5 to 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.

```
# Perform sentiment analysis
english_articles_clean <- english_articles_clean %>%
  mutate(sentiment = get_sentiment(Text, method = "syuzhet"))

# Plot sentiment over time
sentiment_over_time <- english_articles_clean %>%
  group_by(Date) %>%
  summarise(avg_sentiment = mean(sentiment, na.rm = TRUE)) %>%
  ggplot(aes(x = Date, y = avg_sentiment)) +
  geom_line(color = "blue") +
  geom_smooth(method = "loess", span = 0.1, color = "red", size = 0.5) +
  labs(title = "Average Sentiment Over Time",
       x = "Date",
       y = "Average Sentiment") +
  theme_minimal(base_family = "sans", base_size = 11) +
  theme(plot.title = element_text(hjust = 1),
        axis.text.x = element_text(hjust = 1))
sentiment_over_time
```




```
# Add a unique identifier for each row
tokenized_articles <- tokenized_articles %>%
  mutate(document = row_number())

# Sentiment analysis using different lexicons
afinn_sentiments <- tokenized_articles %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(document) %>%
  summarize(afinn_sentiment = sum(value))

bing_sentiments <- tokenized_articles %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(document, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(bing_sentiment = positive - negative)

nrc_sentiments <- tokenized_articles %>%
  inner_join(get_sentiments("nrc"), by = "word",
             relationship = "many-to-many") %>%
  count(document, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
```

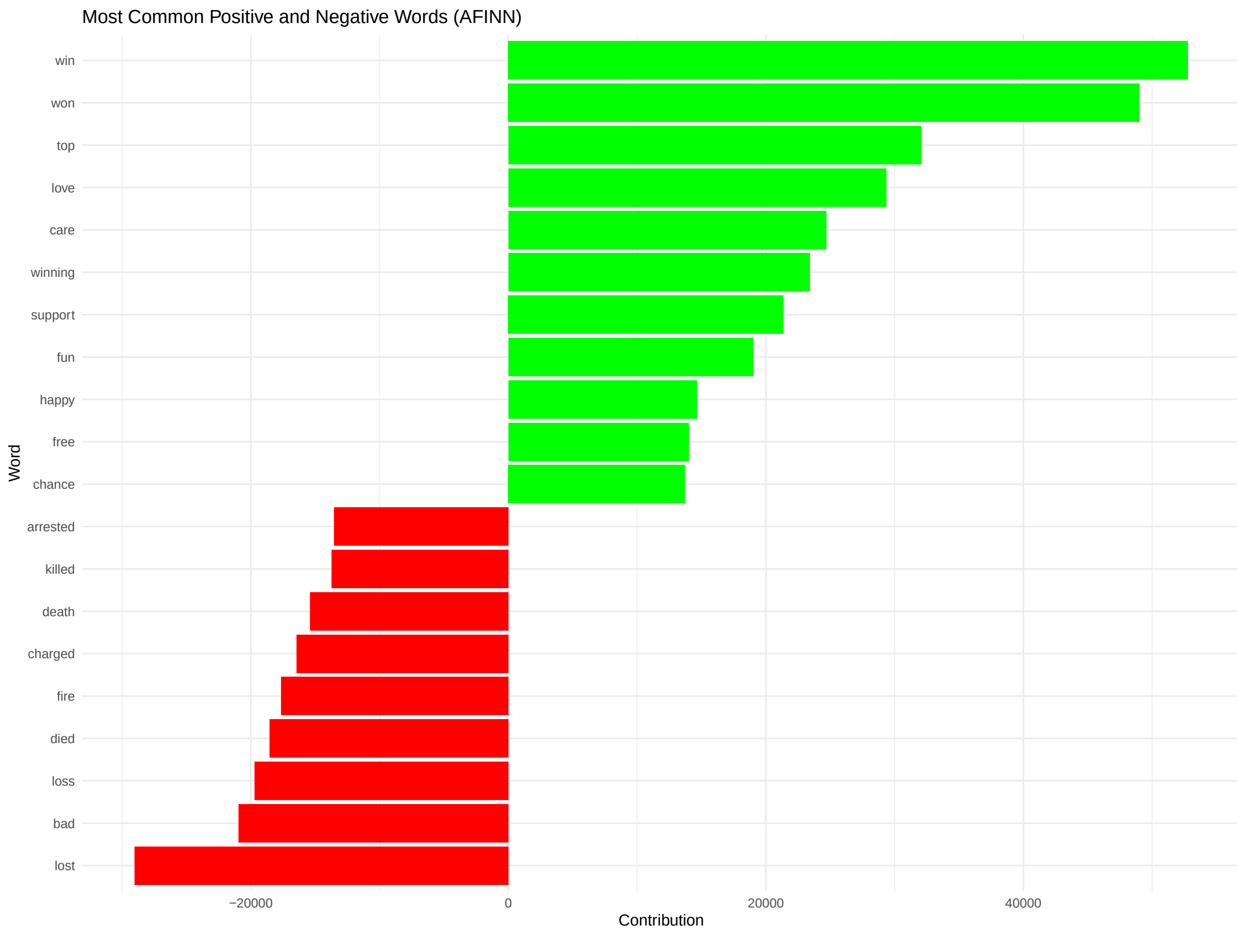
```r
  mutate(nrc_sentiment = positive - negative)

# Compare sentiment lexicons
sentiment_comparison <- english_articles_clean %>%
  mutate(rowid = row_number()) %>%
  left_join(afinn_sentiments, by = c("rowid" = "document")) %>%
  left_join(bing_sentiments, by = c("rowid" = "document")) %>%
  left_join(nrc_sentiments, by = c("rowid" = "document"))

# Save the comparison dataset
write.csv(sentiment_comparison,
          file.path(intermediate_data_dl_path, "sentiment_comparison.csv"),
          row.names = FALSE)

# Plot sentiment comparison
sentiment_comparison_plot <- sentiment_comparison %>%
  gather(key = "lexicon", value = "sentiment", afinn_sentiment,
         bing_sentiment, nrc_sentiment) %>%
  ggplot(aes(x = Date, y = sentiment, color = lexicon)) +
  geom_line() +
  labs(title = "Sentiment Comparison Over Time",
       x = "Date",
       y = "Sentiment") +
  theme_minimal(base_family = "sans", base_size = 11) +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 1)) +
  guides(color = guide_legend(nrow = 1, byrow = TRUE))
sentiment_comparison_plot
```



Next we identify the most common positive and negative words using the AFINN lexicon.

```r
# Identify most common positive and negative words using AFINN lexicon
afinn_contributions <- tokenized_articles %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(word) %>%
  summarize(occurrences = n(),
            contribution = sum(value)) %>%
  arrange(desc(contribution))

# Save the contributions dataset
write.csv(afinn_contributions,
          file.path(intermediate_data_dl_path, "afinn_contributions.csv"),
          row.names = FALSE)

# Plot most common positive and negative words
afinn_contributions_plot <- afinn_contributions %>%
  top_n(20, abs(contribution)) %>%
  mutate(word = reorder(word, contribution)) %>%
  ggplot(aes(x = word, y = contribution, fill = contribution > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
```

```r
  labs(title = "Most Common Positive and Negative Words (AFINN)",
       x = "Word",
       y = "Contribution") +
  scale_fill_manual(values = c("red", "green")) +
  theme_minimal(base_family = "sans", base_size = 11)
afinn_contributions_plot
```

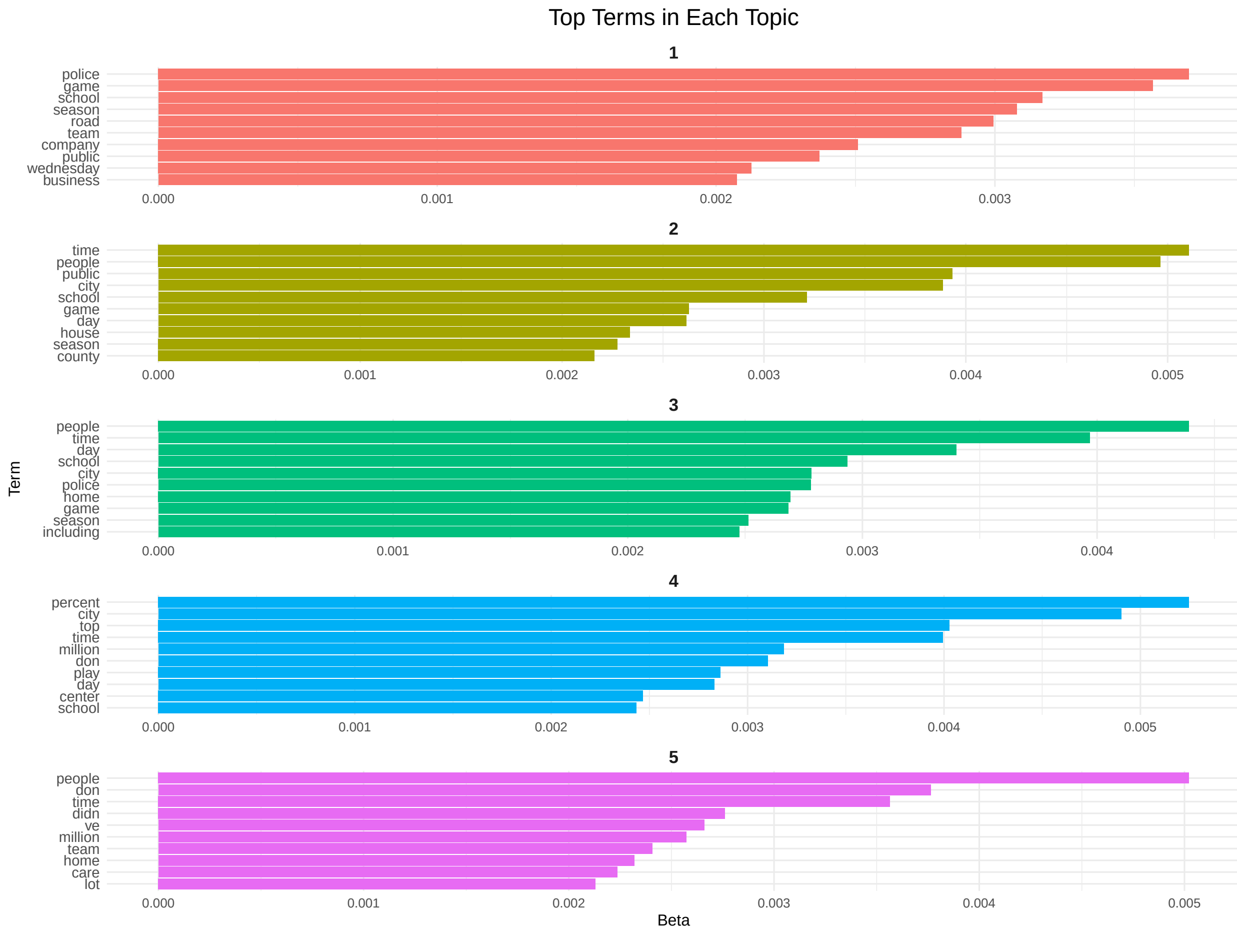


1.3.2 Topic Modeling

Now we apply Latent Dirichlet Allocation (LDA) using the topicmodels package to discover the main topics within the corpus. This model helps us understand the thematic structure of the data across different time periods.

```r
# Create a Document-Term Matrix
dtm <- tokenized_articles %>%
  count(document = 1:nrow(tokenized_articles), word) %>%
  cast_dtm(document, word, n)

# Remove empty documents using the `slam` package
```

```r
dtm <- dtm[slam::row_sums(dtm) > 0, ]

# Apply LDA with k = 5
lda_model <- LDA(dtm, k = 5, control = list(seed = 1234))

# Get top terms for each topic
topics <- tidy(lda_model, matrix = "beta")
top_terms <- topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

# Plot the top terms for each topic
top_terms_plot <- top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free", ncol = 1) +
  coord_flip() +
  scale_x_reordered() +
  labs(title = "Top Terms in Each Topic",
       x = "Term",
       y = "Beta") +
  theme_minimal(base_family = "sans", base_size = 11) +
  theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.text.y = element_text(size = 10),
        strip.text = element_text(size = 12, face = "bold"))
top_terms_plot
```

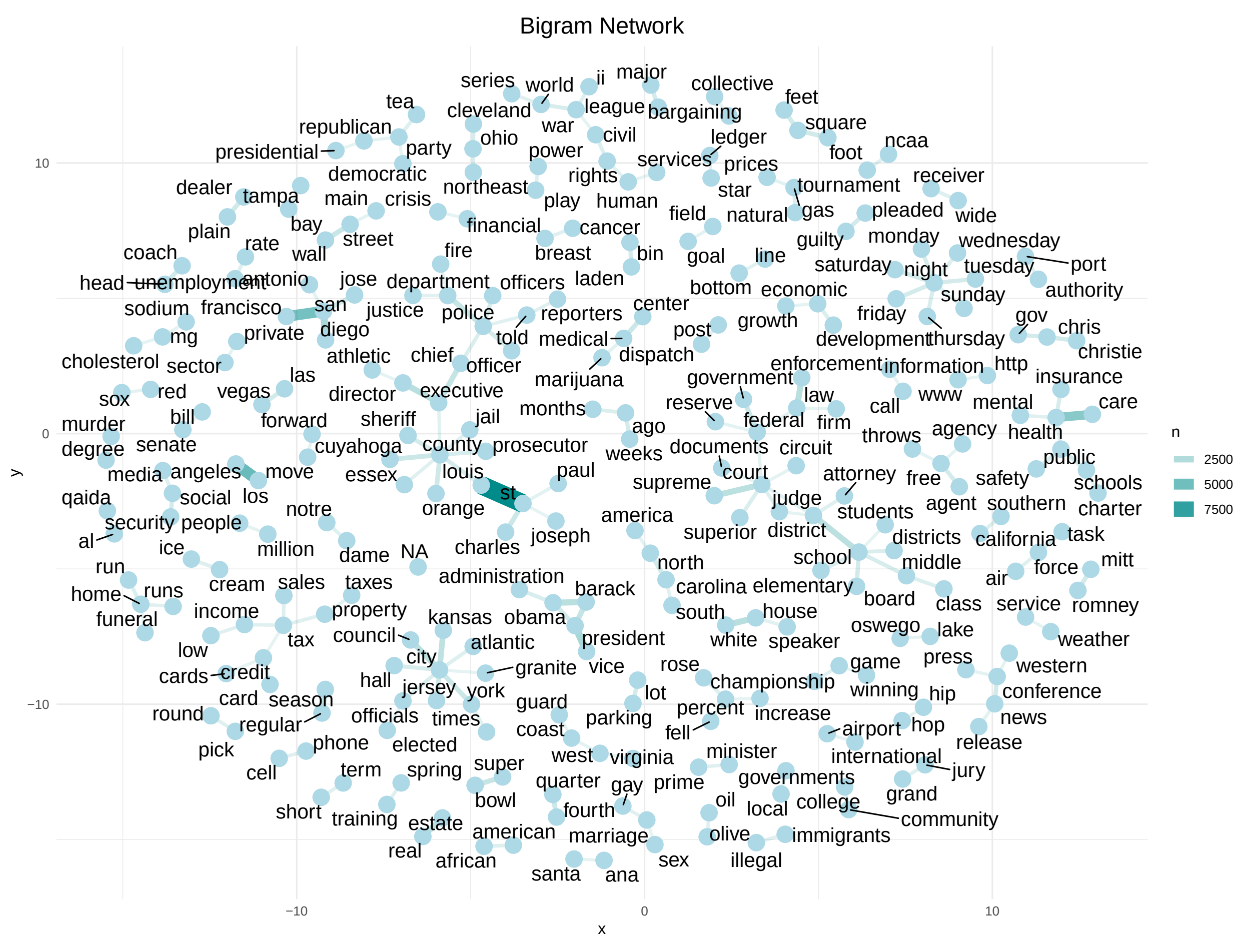


1.3.3 Exploring Relationships Between Words Using N-grams

Using n-grams, we explore the relationships between words in the articles. This involves tokenizing the text into bigrams, filtering out stop words, and visualizing the relationships between commonly co-occurring word pairs.

```
# Tokenize into bigrams
bigrams <- english_articles_clean %>%
  unnest_tokens(bigram, Text, token = "ngrams", n = 2)

# Separate bigrams into two words
bigrams_separated <- bigrams %>%
  separate(bigram, into = c("word1", "word2"), sep = " ")

# Filter out stop words
bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

# Count bigrams
```

```r
bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)

# Create a network plot of bigrams with higher threshold
bigram_graph <- bigram_counts %>%
  filter(n > 500) %>%
  graph_from_data_frame()

# Create a bigram network plot
bigram_plot <- ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "cyan4") +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), size = 5, repel = TRUE) +
  theme_void() +
  labs(title = "Bigram Network") +
  theme_minimal(base_family = "sans", base_size = 11) +
  theme(plot.title = element_text(hjust = 0.5, size = 16))
bigram_plot
```



1.3.4 Word Embeddings

We apply Word2Vec to map words to vector representations, capturing semantic relationships between them using the `text2vec` package. We also visualize these embeddings using t-SNE.

```r
# Prepare the text data
text_data <- english_articles_clean$Text

# Tokenize the text data
tokens <- word_tokenizer(text_data)

# Create iterator over tokens
it <- itoken(tokens, progressbar = FALSE)

# Create vocabulary
vocab <- create_vocabulary(it)

# Create a vectorizer
vectorizer <- vocab_vectorizer(vocab)

# Create the term-co-occurrence matrix (TCM)
tcm <- create_tcm(it, vectorizer, skip_grams_window = 5)

# Fit the GloVe model
glove <- GlobalVectors$new(rank = 50, x_max = 10)
word_vectors <- glove$fit_transform(tcm, n_iter = 20)
```

```
## INFO  [12:00:22.070] epoch 1, loss 0.1850
## INFO  [12:00:31.016] epoch 2, loss 0.1278
## INFO  [12:00:43.128] epoch 3, loss 0.1137
## INFO  [12:00:56.160] epoch 4, loss 0.1064
## INFO  [12:01:11.223] epoch 5, loss 0.1017
## INFO  [12:01:26.408] epoch 6, loss 0.0983
## INFO  [12:01:41.363] epoch 7, loss 0.0958
## INFO  [12:01:56.489] epoch 8, loss 0.0937
## INFO  [12:02:11.676] epoch 9, loss 0.0921
## INFO  [12:02:26.658] epoch 10, loss 0.0906
## INFO  [12:02:41.933] epoch 11, loss 0.0895
## INFO  [12:02:56.922] epoch 12, loss 0.0885
## INFO  [12:03:11.697] epoch 13, loss 0.0875
## INFO  [12:03:26.671] epoch 14, loss 0.0868
## INFO  [12:03:41.489] epoch 15, loss 0.0861
## INFO  [12:03:56.334] epoch 16, loss 0.0855
## INFO  [12:04:11.542] epoch 17, loss 0.0849
## INFO  [12:04:28.592] epoch 18, loss 0.0844
## INFO  [12:04:45.225] epoch 19, loss 0.0839
## INFO  [12:05:00.031] epoch 20, loss 0.0835
```

```r
# Combine main and context vectors into word embeddings
word_vectors <- word_vectors + t(glove$components)

# Find words similar to "economy"
economy_vector <- word_vectors["economy", , drop = FALSE]

# Calculate the cosine similarity between "economy" and all other words
similar_words <- sim2(x = word_vectors, y = economy_vector, method = "cosine",
                      norm = "l2")

# Remove the word "economy" itself and get the top 10 similar words
similar_words <- sort(similar_words[,1], decreasing = TRUE)
similar_words <- similar_words[2:11]  # Exclude the word "economy" itself
similar_words
```

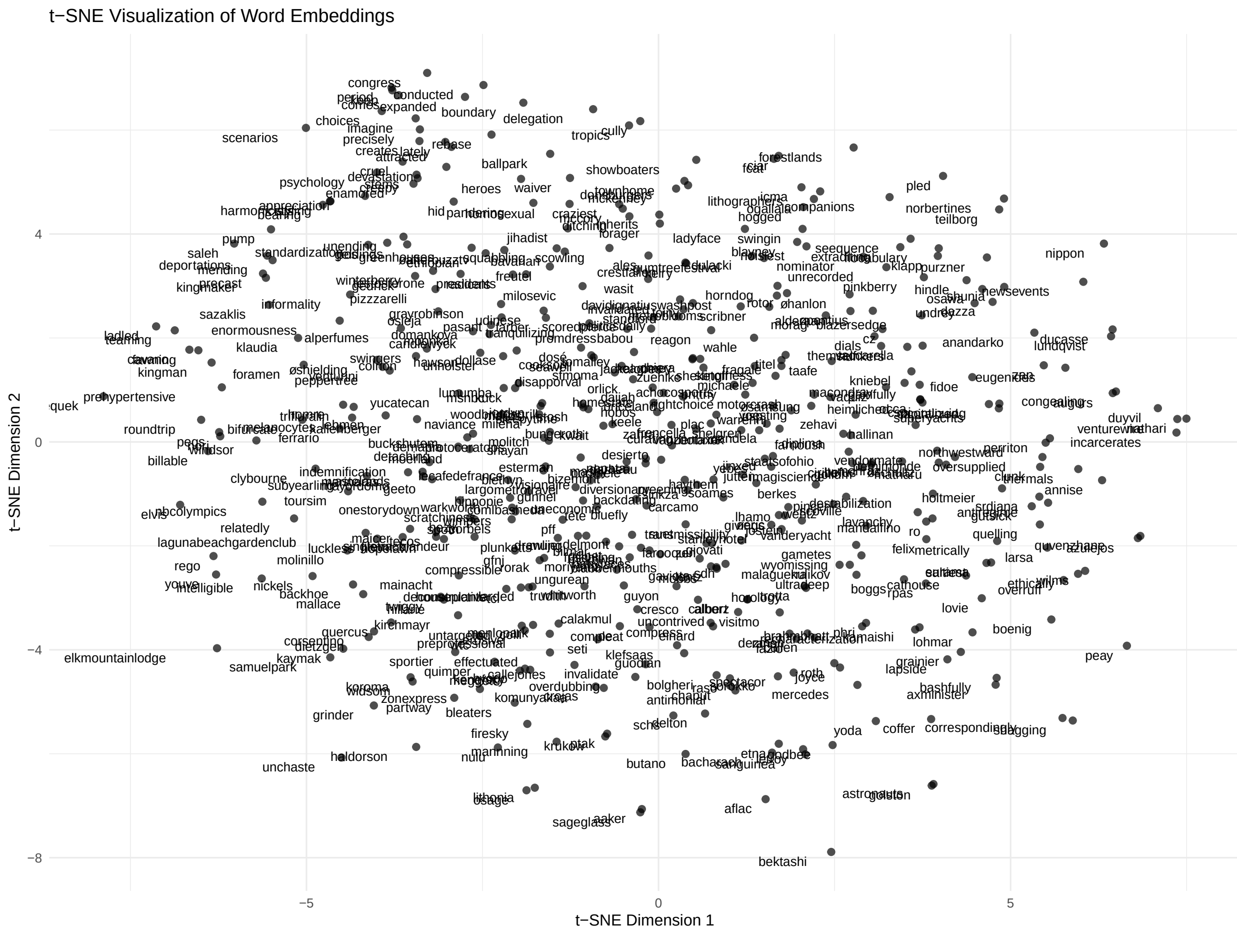
```
##    demand    growth      weak  economic    market    europe   growing recession
## 0.7890692 0.7830211 0.7742456 0.7703297 0.7643266 0.7408833 0.7322566 0.7317352
##   markets    global
## 0.7121498 0.7086140
```

```r
# Save the word vectors for future use
save(word_vectors, file = "word_vectors.RData")

# Load the trained Word2Vec model
load("word_vectors.RData")

# Shuffle the word vectors randomly
set.seed(123)  # For reproducibility
word_vectors <- word_vectors[sample(nrow(word_vectors)), ]

# Get a random sample of 500 word vectors for visualization
words_to_visualize <- word_vectors[1:500,]
word_labels <- rownames(words_to_visualize)

# Perform t-SNE
tsne_model <- Rtsne(words_to_visualize, dims = 2, perplexity = 30,
                    verbose = TRUE, max_iter = 500)
```

```
## Performing PCA
## Read the 500 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.17 seconds (sparsity = 0.260320)!
## Learning embedding...
```

```
## Iteration 50: error is 61.504103 (50 iterations in 0.13 seconds)
## Iteration 100: error is 63.535185 (50 iterations in 0.14 seconds)
## Iteration 150: error is 63.387907 (50 iterations in 0.13 seconds)
## Iteration 200: error is 63.014448 (50 iterations in 0.12 seconds)
## Iteration 250: error is 61.326390 (50 iterations in 0.13 seconds)
## Iteration 300: error is 2.089173 (50 iterations in 0.07 seconds)
## Iteration 350: error is 2.022628 (50 iterations in 0.06 seconds)
## Iteration 400: error is 2.000547 (50 iterations in 0.06 seconds)
## Iteration 450: error is 1.987653 (50 iterations in 0.06 seconds)
## Iteration 500: error is 1.984894 (50 iterations in 0.05 seconds)
## Fitting performed in 0.95 seconds.
```

```r
tsne_data <- as.data.frame(tsne_model$Y)
tsne_data$word <- word_labels

# Plot the t-SNE results
tsne_plot <- ggplot(tsne_data, aes(x = V1, y = V2, label = word)) +
  geom_point(size = 2, alpha = 0.7) +
  geom_text(aes(label = word), size = 3, vjust = 1.5, hjust = 1.5) +
  theme_minimal(base_family = "sans", base_size = 11) +
  labs(title = "t-SNE Visualization of Word Embeddings",
       x = "t-SNE Dimension 1",
       y = "t-SNE Dimension 2")
tsne_plot
```



2 Results

2.1 Temporal Distribution

The analysis of the temporal distribution of articles revealed significant trends in the data. The number of articles published increased steadily over the years, with a notable spike in articles from 2010 onwards. This trend likely corresponds to the increased digitalization and online availability of newspaper archives. When examining the number of articles by year, it becomes evident that the increase in articles is particularly pronounced in 2012. This significant rise can be attributed to better archival practices and improved accessibility of digital archives during this period. We analyzed the frequency of words in the articles to understand the most commonly used terms. The top three most common words are “time,” “people,” and “city.”

2.2 Sentiment Analysis

2.2.1 Average Sentiment Over Time

Using the `syuzhet` package, we analyzed the average sentiment of articles over time. The average sentiment fluctuated significantly throughout the period, indicating a wide range of emotional tones in the articles. There was a notable stabilization in sentiment scores from 2008 onwards, suggesting more consistent reporting or less emotional variability in later years.

2.2.2 Lexicon Comparison

We compared sentiment scores using different lexicons (`AFINN`, `bing`, and `nrc`). The comparison revealed that each lexicon captured different aspects of sentiment, with some overlap. The `AFINN` lexicon identified more extreme sentiments, while the `bing` and `nrc` lexicons provided a balanced view of positive and negative sentiments.

2.2.3 Most Common Positive and Negative Words (AFINN)

We identified the most common positive and negative words using the AFINN lexicon. The most frequent positive words included `win`, `top` and `love`, while common negative words were `lost`, `bad` and `loss`.

2.3 Topic Modeling

We applied Latent Dirichlet Allocation (LDA) to identify the main topics within the corpus. The LDA model with 5 topics revealed distinct themes with terms such as police, time, people, percent, and school in different contexts. Each topic was characterized by its top terms, which provided insights into the prevalent subjects discussed in the articles.

2.4 Exploring Relationships Between Words Using N-grams

By analyzing bigrams, we explored the relationships between words in the articles. The bigram network plot revealed frequently co-occurring word pairs, highlighting common phrases and terms used in the articles.

2.5 Word Embeddings

We used the `text2vec` package to create word embeddings and visualized these embeddings using t-SNE (t-distributed Stochastic Neighbor Embedding).`t-SNE` is a dimensionality reduction technique that helps to visualize high-dimensional data in a lower-dimensional space (usually 2 or 3 dimensions). It is particularly well-suited for embedding data for visualization because it preserves the local structure of the data. The t-SNE plot shows a 2D visualization of word embeddings, where each point represents a word, and the distances between points reflect their semantic similarities.

3 Conclusion

3.1 Summary of Findings

This project analyzed the Old Newspapers dataset to uncover trends, sentiments, and relationships between words over time. The temporal distribution analysis showed a significant increase in articles from 2010 onwards. Sentiment analysis revealed fluctuating emotions in earlier years, stabilizing over time. Topic modeling identified key themes in the articles, and n-gram analysis highlighted common word pairs.Additionally, word embeddings provided insights into the semantic relationships between words.

3.2 Potential Impact

The insights gained from this analysis can be valuable for historians, sociologists, and researchers interested in understanding historical trends and societal changes through newspaper articles. The sentiment analysis and topic modeling techniques can be applied to other corpora to uncover similar insights.

3.3 Limitations

There are several limitations to this study. The analysis was conducted on articles written in English, which may limit the generalizability of the findings to other languages within the dataset. Additionally, the dataset contains poor data from earlier years, which might affect the accuracy of trend analysis. The sentiment analysis and topic modeling techniques also have inherent limitations, such as potential biases in the lexicons and the need for manual tuning of model parameters.

3.4 Future Work

Future work could involve expanding the analysis to include articles in other languages and applying more advanced machine learning techniques, such as clustering, to uncover additional insights. Additionally, integrating other data sources, such as social media posts or government records, could provide a more comprehensive view of historical trends and sentiments. Exploring more sophisticated models and methods to improve sentiment analysis and topic modeling can also enhance the depth and accuracy of the findings.

References

- AFINN lexicon from Finn Årup Nielsen. URL: https://github.com/fnielsen/afinn
- Alvations, S. (2019). Old Newspapers Dataset. Kaggle. Retrieved from https://www.kaggle.com/datasets/alvations/old-newspapers
- bing lexicon from Bing Liu and collaborators. URL: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
- Jockers, M. (2021). Syuzhet: Extract Sentiment and Plot Arcs from Text. R package version 1.0.6. https://cran.r-project.org/web/packages/syuzhet/index.html
- nrc lexicon from Saif Mohammad and Peter Turney. URL: http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm
- Silge, J., & Robinson, D. (2017). Text Mining with R: A Tidy Approach. O'Reilly Media, Inc. URL: https://www.tidytextmining.com/
- The complete project, including all code and data, is available on GitHub: https://github.com/krotov79/oldnewspapers