CSCA 5028 - Final Project - Data and Text Analysis

by Joan Kusuma

Weekly Article Releases and Claps

This function calculates the number of articles released and the sum of claps per week for a specified publisher. It groups the data by week using `published_date` to count releases and sum claps.

$$\text{releases\_by\_week} = \sum_i 1_{\text{week}(d_i)=w} \quad \text{and} \quad \text{claps\_by\_week} = \sum_{i \in \text{week}(d_i)=w} \text{claps}_i$$

Here:

- $\text{week}(d_i) = \text{w}$ represents grouping by each week w based on the published $\text{date}(d_i)$.
- $1_{\text{week}(d_i)=w}$counts the number of articles per week w for releases.
- $\sum_{i \in \text{week}(d_i)=w} \text{claps}_i$gives the total claps in week w by summing the claps for all articles published that week.

Releases vs Claps by Day of the Week

This function calculates the average number of articles published and the average number of claps per day for a given publisher. It groups the articles by `pub_day` (the day of the week) and calculate: * The average number of articles published per day by counting the `title` column * The average number of claps per day by averaging the `claps` column

It then devides the number of articles published by the number of unique weeks to get the correct daily average

$$\text{avg\_articles\_published\_by\_day} = \frac{\sum_{\text{articles on day } d} 1_{\text{published\_on\_day}(d)}}{\text{unique\_weeks}}$$

$$\text{avg\_claps\_per\_day} = \frac{\sum_{\text{articles on day } d} \text{claps}_i}{\text{unique\_weeks}}$$

Where:

- $1_{\text{published\_on\_day}(d)}$ counts the number of articles published on day d.

Claps distribution of each publisher

This function calculates the distribution of log-transformed claps for articles by publisher. To do that, I'm going to first calculate the outliers, and group the data by publishers and calculate statistics for log-transformed claps, such as, min Q1, median, Q3, max.

$$\text{Outliers} = \{x \in \text{series} \mid x < \text{Lower bound or } x > \text{Upper bound}\}$$

Where:

- Q_1 = 25th percentile of series
- Q_3 = 75th percentile of series
- IQR = Q_3 - Q_1
- Lower bound = Q_1 - 1.5 × IQR
- Upper bound = Q_3 + 1.5 × IQR

Finally:

Log claps distribution statistics per collection:

- Minimum = min(log claps)
- Q_1 = 25th percentile
- Median = 50th percentile
- Q_3 = 75th percentile
- Maximum = max(log claps)
- Outliers = $\{x \in \text{log claps} \mid x < \text{Lower bound or } x > \text{Upper bound}\}$

Publisher's Articles Count

This function calculates the total number of articles published by each publisher

Number of articles per collection = Count of articles for each publisher's collection

Unique Authors per Collection

This function calculates the number of unique authors for each publisher's collection

Unique authors per collection = Count of unique authors for each collection

N-Gram Frequency

This function calculates the top 20 n-grams based on frequency in the article titles. It does this by tokenizing the titles, generate n-grams, and calculate the frequency of each n-gram. For the analysis, I did bigram and trigrams. They are also done for articles that have higher engagement by looking at the number of claps and can be compared to see how higher engaging articles are performing compared to all articles.

All Articles n-gram formula:

Top 20 n-grams by frequency = {keywords: count | n-gram appears in titles}

n-gram formula for articles with higher engagement:

- Average claps $= \frac{\sum \text{Claps}}{\text{Total articles}}$

Latent Dirichlet Allocation (LDA) Model

LDA is a generative probabilistic model that aims to discover latent topics in a collection of documents. Each document is assumed to be generated by a mixture of topics, where each topic is a distribution over words.

1. Tokenization Tokenization is the process of breaking down a document into individual tokens (words):

$$\text{tokens}_i = \text{Tokenize}(\text{title\_cleaned}_i)$$

Where:

- title_cleaned_i is the cleaned title of the i-th article.
- Tokenize refers to the process of splitting the cleaned title into individual tokens.

Example: if the document is "dog runs fast", tokenization splits it into three tokens: ["dog", "runs", fast]

2. Creating the Dictionary The dictionary is built from all unique tokens in the corpus. It is created by taking the union of all the words from every document (tokens). This means each word that appears in any document is added to the dictionary, but only once.

$$V = \bigcup_{i=1}^{N} \text{tokens}_i$$

Where:

- V is the vocabulary, the set of all unique words.
- N is the total number of documents.

Example: if the documents are ["dog runs fast", "cat runs fast"], the dictionary will be ["dog", "runs", "fast", "cat"]

3. Creating the Corpus The corpus is a collection of BoW (Bag-of-Words) vectors, where each document is represented by the frequency of words from the vocabulary:

$$b_i = [f(w_1, \text{doc}_i), f(w_2, \text{doc}_i), \ldots, f(w_k, \text{doc}_i)]$$

Where:

- $f(w_j, \text{doc}_i)$ is the frequency of word w_j in document doc_i.
- k is the total number of unique words in the vocabulary.

Example:

Ken has three documents:

- Document 1: "dog runs fast"
  - "dog" appears 1 time
  - "runs" appears 1 time
  - "fast" appears 1 time
  - "cat" appears 0 time
  - So, the vector for document 1 is: [1, 1, 1, 0]
- Document 2: "cat runs fast"
  - "dog" appears 0 time
  - "runs" appears 1 time
  - "fast" appears 1 time
  - "cat" appears 1 time
  - Vector for document 2 is: [0, 1, 1, 1]
- Document 3: "dog cat"
  - "dog" appears 1 time
  - "runs" appears 0 time
  - "fast" appears 0 time
  - "cat" appears 1 time
  - Vector for document 3 is: [1, 0, 0, 1]

Now, the corpus is simply a matrix where each row represents a document and each column corresponds to a word from the dictionary. The matrix (BoW corpus) is the input to the LDA model, which will analyze these frequencies to find underlying topics.

- **Matrix Form**:

Document	"dog"	"runs"	"fast"	"cat"
Doc 1	1	1	1	0
Doc 2	0	1	1	1
Doc 3	1	0	0	1

4. Latent Dirichlet Allocation (LDA) Model The LDA model assumes each document is a mixture of topics, where each topic has a specific distribution over words, and each document has a mixture of these topics. The goal is to uncover the underlying topics and assign a distribution of words to each topic, based on the documents in the corpus.

The Likelihood Function:

The likelihood function in LDA expresses the probability of the observerd words in a document given the topic and word distribution:

$$P(w, z|\alpha, \beta) = \prod_{d=1}^{D} P(w_d|\alpha, \beta)$$

Where:

- w_d represents words in document d.
- α and β are hyperparameters that control topic and word sparsity.

This function is maximized using methods like Gibbs sampling or Variational Inference.

Example: Given documents about dogs and cats, LDA learns topics like:

- Topic 1: "Animals" (dog, cat)
- Topic 2: "Actions" (runs, fast)
- Document 1 : "dog runs fast"
- Document 2: "cat runs fast"

LDA estimates 80% of Document 1 relates to Topic 1 (Animals) and 20% to Topic 2 (Actions).

The 80% comes from the topic distribution for Document 1. LDA doesn't just count words; it probabilistically estimates that 80% of the document's content is about Topic 1 (Animals), even if not all words are directly linked to that topic. For example, the word "dog" in Document 1 has a 60% probability of being from Topic 1 (Animals) and 10% from Topic 2 (Actions)

After iterative updates, LDA adjusts these distributions to best match the observed data, maximizing the likelihood function. In practice, when training an LDA model, the goal is to maximize the likelihood or equivalently minimize the negative log-likelihood.