DSC210: Non-Negative Matrix Factorization (NMF) vs. BERTopic for Topic Modeling

FA'24 | December 5, 2024

Sarah Borsotto
MS, Data Science
A16209665

John Driscoll
MS, Data Science
A16298297

Taylor Martinez
MS, Data Science
A69034713

Thuy Nguyen
MS, Data Science
A69034506

1 Introduction

With over 402.89 million terabytes [23] of data generated daily—80 percent of which is unstructured [3]—businesses face significant challenges in analyzing and deriving value from this data. While unstructured data is intuitive for humans to interpret, it is difficult for machines to process efficiently. Topic modeling addresses this gap by offering an automated way to summarize and organize unstructured information.

Topic modeling can be defined as an unsupervised machine learning technique used to uncover the underlying themes associated with a collection of documents by identifying semantic patterns and summarizing the prominent subjects within the text. It simplifies large text datasets by grouping similar words into clusters representing distinct topics.

Topic modeling has diverse applications, including automatically tagging customer support tickets and detecting urgency [18], categorizing news articles by topic, analyzing customer reviews [14], and more, which makes it a valuable tool for gaining actionable insights from text datasets.

In this project, we compare and evaluate the performances of two approaches – the algebraic model, non-negative matrix factorization (NMF), and the more modern BERTopic model – by applying them to the 20-NewsGroup dataset.

1.1 History/Background

The origins of topic modeling trace back to 1990, when Deerwester introduced Latent Semantic Analysis (LSA), an early attempt to address the task of automating document indexing and retrieval [4]. LSA leverages Singular Value Decomposition (SVD) to identify the most important terms in documents, making it the first systematic approach to discern themes in text. Lee simultaneously developed NMF as an alternative approach for factorizing document-term matrices to extract topics [4].

In 2003, Blei introduced Latent Dirichlet Allocation (LDA), which utilizes a Bayesian approach. This method quickly grew in popularity due to its flexibility and ability to be extended to more complex models. These early topic modeling techniques were primarily designed to analyze stable and structured document corpora [4].

With the rise of social media and real-time text generation, topic modeling methods had to adapt to handle shorter, noisier text and incorporate temporal dynamics. This included using transformers,

a deep learning attention mechanism, leading to increased accuracy and efficiency. Newer techniques like BERTopic use pre-trained embeddings to capture semantic context, providing greater interpretability and performance than the more traditional methods.

1.2 Applications

Topic modeling has numerous real-world applications and plays a critical role in interpreting and organizing unstructured data into meaningful clusters. It acts as the foundation of many natural language processing (NLP) tasks, including information retrieval, trend analysis, content recommendation, and sentiment analysis. [17]

For example, topic modeling helps search engines understand the intent behind a user's query and deliver more relevant results. Similarly, for social media, streaming platforms, and e-commerce sites, it identifies themes within user-generated content to recommend personalized content or products by recognizing patterns in user behavior and predicting what they would likely be interested in. By analyzing the frequency and contextual relevance of words, topic modeling effectively categorizes and prioritizes information, streamlining data-driven decision-making across several domains.

A real world example: it can show how word co-occurrence and weighted bi-graph clustering approaches can be used to represent how a brand like Lipton shows up in search queries and what topics are related to the clusters.

cluster	**queries**
nutrition and calories	calories in lipton tea how many calories in lipton tea lipton black tea nutrition facts lipton tea bag calories lipton tea bags nutrition facts lipton tea nutrition
weight loss	can lipton green tea help lose weight does lipton green tea burn fat diet lipton green tea weight loss lipton green tea benefits weight loss
company information	history of lipton tea home country of lipton tea lipton tea origin lipton tea history
k cups (keurig)	keurig lipton iced tea lipton iced tea k cups lipton k cups lipton sweet tea k cup

Table 3: Example semantic clusters for Lipton brand queries using bigraph clustering.

Figure 1: Lipton Bigraph Clustering [22]

1.3 State-of-the-art

Topic modeling has evolved significantly over the past few decades, with both traditional and modern approaches serving as powerful tools for identifying themes in text data. Our analysis focuses on two key approaches: NMF and BERTopic.

NMF is a traditional, linear-algebra-based machine learning technique with the primary goal of decomposing a term-document matrix into two smaller non-negative matrices representing topics and word associations. These matrices ultimately help in identifying patterns by grouping words that frequently appear together. This is a straightforward approach that is generally easy to implement, but it struggles with more complex data [10].

On the other hand, BERTopic is a modern, transformer-based approach that relies on BERT embeddings to capture the semantic meaning of documents before applying dimensionality reduction and clustering to identify patterns within the text. This technique excels at analyzing large and complex datasets with high accuracy, but it is much more computationally intensive [10].

2 Problem Formulation

2.1 Relation to Numerical Linear Algebra

Term frequency–inverse document frequency, also known as TF-IDF, is an NLP technique that is used to measure the importance of words in a corpus.

TF-IDF is composed of two components - term frequency and inverse document frequency. Term frequency refers to the number of times a specific word appears in a document [**?**]. This is calculated using the equation:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1}$$

where the numerator $f_{t,d}$ represents the number of times a term t can be found within a document d, and the denominator represents the total number of terms within the document [11].

Example: For this TF-IDF example, we will use these two sentences.[12]

Sentence 1: The car is driven on the road

Sentence 2: The truck is driven on the highway

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Figure 2: A chart showing TF-IDF breakdown of two sentences listed above

We first calculate TF by calculating the amount of times a word appears and diving it by the total number of words. (ex. 'car' appears once within the 7 word first sentence 'A', so it is labeled '1/7')

We then calculate IDF by taking the logarithm of the total number of sentences, and then diving that by the number of sentences containing that specific word. (ex. 'car' appears once in a total of two sentences, so we take log (2/1)

Finally, to get the TF-IDF, we multiply the values we get from the corresponding TF sentence and IDF together to get the corresponding TF-IDF value.
(ex. '1/7' (0.143) multiplied by log (2/1) (0.3) = 0.043 for the first sentence A's TF-IDF's score, '1/7' (0.143) multiplied by log (2/2) (0) = 0 for the second sentence B's TF-IDF' score).

We now have a document-word matrix, where each row is a document, each column is a unique word, and each entry is the TF-IDF score for the word in that document. This highlights important and unique words in each document and represents the documents as vectors, which can be clustered or decomposed into topics.

2.2 Approach Description for NMF

Non-negative matrix factorization (NMF) is an unsupervised learning algorithm used to extract topics from a collection of documents. NMF decomposes a non-negative matrix into two non-negative matrices that represent the topic distribution across documents and the word distribution across topics. This decomposition is useful for identifying the underlying topics within the document collection [9].

In the context of natural language processing (NLP), the input matrix $\mathbf{A}$ is a document-word matrix in which rows represent documents and columns represent words. A weighted frequency (typically TF-IDF) is used for each term. NMF factorizes $\mathbf{A}$ into two smaller, non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ where:

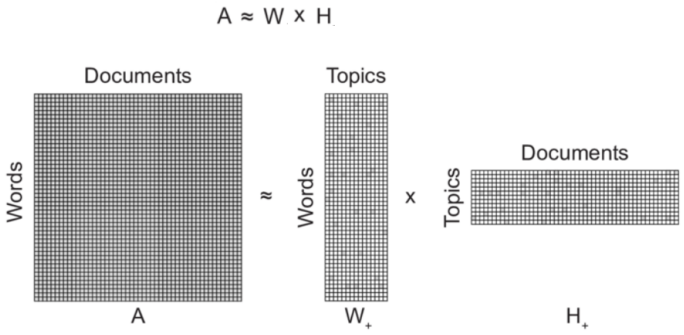


Figure 3: Matrix Factorization [9]

- $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the document-word matrix (with m documents and n terms).
- $\mathbf{W} \in \mathbb{R}^{m \times k}$ is the document-topic matrix where each row represents the distribution of topics in a document.
- $\mathbf{H} \in \mathbb{R}^{k \times n}$ is the topic-word matrix where each column represents the distribution of words in a topic.
- k is the number of topics (selected beforehand) [13].

The goal of NMF is to find non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ whose product is as close as possible to the original input matrix $\mathbf{A}$. Since NMF is unsupervised, we typically use the Kullback-Liebler divergence or the Frobenius norm to minimize this difference.

The Frobenius norm (calculated by taking the square root of the sum of the absolute squares of its elements) measures the difference between $\mathbf{A}$ and $\mathbf{WH}$ [9]:

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (|a_{ij}|)^2} \tag{2}$$

The Kullback-Liebler divergence is a statistical method used to describe how different distributions are from one another. A smaller divergence implies more closeness between words [9]:

$$\text{kl\_div}(x, y) = \begin{cases} x \log\left(\frac{x}{y}\right) - x + y, & \text{if } x > 0, y > 0, \\ y, & \text{if } x = 0, y \geq 0, \\ \infty, & \text{otherwise.} \end{cases}$$

The optimization process for an NMF model often involves iterative algorithms. These multiplicative updates are meant to reduce the error at each iteration while ensuring that $\mathbf{W}$ and $\mathbf{H}$ remain non-negative. The two most common are:

- Coordinate Descent Solver: optimizes one element of $\mathbf{W}$ or $\mathbf{H}$ at a time while keeping the other fixed.
- Multiplicative Update Solver: Updates $\mathbf{W}$ and $\mathbf{H}$ using multiplicative rules [7].

After training the NMF model, the topics represented in $\mathbf{H}$ can be analyzed by looking at the words with the highest weights in each topic. To assess the quality of the topics, topic coherence scores are often used. These scores measure how frequently the top words in each topic appear together in the corpus, with higher coherence indicating that the words form a more interpretable and meaningful topic.

For our experiment, we preprocess the 20-NewsGroup dataset before performing NMF to extract topics.

2.3 SOTA approach description

BERTopic is a transformer-based machine learning model that uses Bidirectional Encoder Representations from Transformers (BERT) embeddings to perform topic modeling. The process of implementing a BERTopic model consists of five main steps [19].

The first step is preprocessing, where the titles and keywords of the documents are encoded into embeddings using pre-trained BERT models. These embeddings are high-dimensional vectors that represent the semantic meaning of the text, providing a high-dimensional representation of the documents [6].

Then, the Uniform Manifold Approximation and Projection (UMAP) algorithm is applied to reduce the dimensionality of the data through the computation of pairwise distances and stochastic gradient descent, using graph theory and matrix operations to preserve high-dimensional relationships [25].

Next, the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm is used to group similar data points together based on density and identify clusters of related documents while ignoring outliers. To estimate density, the mutual reachability distance is calculated using the equation:

$$d_{\text{mreach-}k}(a, b) = \max\left\{\text{core}_k(a), \text{core}_k(b), d(a, b)\right\}$$

Where $d(a, b)$ represents the original metric distance between a and b [1].

In the fourth step, the class-based Term Frequency-Inverse Document Frequency (c-TF-IDF) algorithm is used to analyze the importance of words in each cluster by looking at term frequency within each cluster and their frequency across different clusters.

Lastly, we visualize and interpret the results.

For our experiment, we follow these steps to apply BERTopic to the 20-NewsGroup dataset and define interpretable topics.

2.4 Theoretical Comparison of BERTopic and NMF

BERTopic and Non-Negative Matrix Factorization (NMF) are both topic modeling techniques with distinct theoretical underpinnings. BERTopic leverages pre-trained embeddings to capture the semantic relationships between words and documents, enabling it to identify nuanced and context-aware topics. It combines dimensionality reduction methods like UMAP with clustering algorithms like HDBSCAN to group similar documents into topics based on their embeddings. In contrast, NMF is a linear algebra-based approach that decomposes a document-term matrix into two non-negative matrices: a topic-term matrix and a document-topic matrix. This decomposition represents each document as a combination of topics and each topic as a combination of terms, relying on word frequency and co-occurrence patterns in the corpus. While BERTopic focuses on contextual

embeddings for enhanced semantic understanding, NMF emphasizes word-level patterns in the textual data.

NMF and BERTopic differ in their computational demands, ease of use, and output characteristics. NMF is computationally efficient, easy to implement, and requires the user to define the number of topics in advance. It uses matrix operations, which can sometimes lead to incoherent topics, and each document can belong to multiple topics with associated probabilities. In contrast, BERTopic is computationally intensive but allows for multilingual analysis and eliminates the need for extensive preprocessing by using embeddings. It automatically determines the number of topics, can prune topics (with potential fidelity loss), and assigns each document to a single topic without providing probabilities for multiple topics [8]. We will be comparing these models analytically in our two experiments.

2.5 Evaluation

Evaluating selected topics is inherently challenging, as different algorithms uncover varying topics that best represent the documents, and no single set of topics can be deemed universally "best." Individual preferences also play a role—some topic models may resonate more with users because they align closely with their cognitive frameworks, a subjective quality that is difficult to quantify. While certain metrics attempt to capture this subjectivity, when the goal is to enhance human understanding rather than simply reduce dimensionality, it can be beneficial for topic model developers to engage experts for qualitative evaluation.

Topic Coherence

Topic coherence measures the degree to which the words within a topic are related to each other, based on the documents in the training set. Röder et al. (2015) [5] describe coherence measures as a pipeline consisting of four key steps: segmentation, probability calculation, confirmation measure, and aggregation.

1. **Segmentation** splits the topics into subsets, such as word pairs.
2. **Probability calculation** assesses the likelihood that a word or word pair appears in the documents.
3. **Confirmation measure** quantifies the likelihood that these word subsets co-occur within the documents.
4. **Aggregation** combines these measures into a final coherence score.

Topic coherence can be compared across different topic models since it only requires the generated topics and the training corpus. In our implementation, the probability calculation and confirmation measure are performed by assessing the similarity between word pairs using word2vec [15] embeddings of the training documents.

Topic Diversity

Topic Diversity measures how distinct topics are from one another. The more diverse the topics, the more they will cover the different themes of the corpus. [24] Some metrics to measure the diversity of topics include the proportion of unique words across topics, the inverse of their Jaccard similarity, and using embeddings to measure their distance in vector space. We measure diversity by the proportion of unique words for the top 10 words in each topic.

Visualization

Visualizations provide a more intuitive understanding of topics than simply presenting them as a list of words. Common visualizations include word clouds, which highlight the most prominent words within a topic, making it easier to identify key terms. Topics can also be converted into embeddings and plotted on two-dimensional axes after dimensionality reduction. If topics are well spread along the principal components, it suggests greater distinctness between them. Lastly, a barchart can depict the relationships between the topic distributions. The BERTopic model has a class function that maps documents to their dominant topic and counts the corresponding newsgroups. We can use this to see

how well our topic model maps back to the original themes. We implemented a similar graph for the NMF model [16].

Expert Evaluation

Subject matter experts (SMEs) can manually evaluate the relevance of discovered topics by assigning scores to individual topics, providing descriptions of their relevance, or approving or rejecting groups of topics as a whole. SMEs can also enhance the interpretability of the output by manually assigning descriptive labels to each topic. An emerging area of research involves using large language models (LLMs) to automate this labeling process. In our evaluation, we manually inspect a subset of topics and assess whether we can discern a clear theme or concept that each topic is attempting to describe.

Residual Evaluation

An evaluation metric unique to NMF and other topic models that output decomposed matrices is the model residual [21]. This metric addresses the question: *How well do the topics approximate the original tf-idf matrix weights for all documents?* The residual is calculated by taking the Frobenius norm (the square root of the sum of the absolute squares of the matrix elements) of the difference between the original tf-idf matrix and the dot product of the decomposed matrices W and H. Specifically, it measures the difference between the estimated topic weights and the actual values to assess the quality of the approximation across all topics:

$$||A||_f = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(|a_{ij}|)^2} \tag{3}$$

where each document is s the i-th row of the document-topic matrix W and H is the topic-term matrix [20].

Although descriptive and easily calculated, residual is difficult to interpret because it cannot be compared across different types of models.

3 Experiments

3.1 Setup and logistics

In this project, we utilized GitHub to store and share our code, which was developed in Python and executed in a Jupyter Notebook environment.

Utilities: We used pandas and numpy for data handling, and matplotlib and seaborn for plotting.

Data Processing: We used ENGLISH_STOP_WORDS to remove common stop words, TweetTokenizer to tokenize text into meaningful components, WordNetLemmatizer to reduce words to their base forms, nltk for stemming.

Model Building and Evaluation: We used TfidfVectorizer to transform text into numerical representations based on term frequency-inverse document frequency. We applied sklearn's NMF and standard BERTopic as the base for topic modeling. We supplemented the packages with TSNE for dimensionality reduction, UMAP to uncover complex structures in data, HDBSCAN for clustering, gensim's word2vec for evaluation, tqdm for progress bars, and octis for diversity measurement.

3.2 Dataset and programming

Our dataset is the 20newsgroups dataset, a collection of 18,000 newsgroup documents from 20 different newsgroups [2]. This dataset is a common benchmark for document classification and topic modeling tasks. We train our topic models on a subset of the training set and extract 10 of the most relevant topics to make the original topics more digestible. We use the testing labels to sanity check that our methods are capturing relevant information during training.

The newsgroups we retain after pruning and their corresponding counts of documents are below:

newsgroup	count
rec.sport.hockey	600
soc.religion.christian	599
rec.motorcycles	598
sci.crypt	595
sci.med	594
sci.space	593
misc.forsale	585
comp.graphics	584
talk.politics.mideast	564
talk.politics.guns	546

The total number of observations in this subset are 5858. We investigate the contents of these documents and find that the mean number of words is 348 with a standard deviation of 725. The maximum number of words is 20235 and the minimum is 11. There are 1009834 distinct words in the corpus. As we can see, there are large deviations per document and there are over a million unique terms in our collection of documents. We want our model to extract meaningful themes from our data, and we need to do this efficiently, so we preprocess the data to only include necessary tokens. We first clean the documents by removing punctation, numbers, and characters, converting to lowercase, stripping empty strings, and deleting tokens smaller than 2 characters. We also remove non-descriptive words such as "a", "is", "from", "and", etc. We add additional stop words that occur often in our dataset but do not have any significant meaning for the theme extraction. Lastly, we lemmatize our documents, meaning that we distill words into their foundational terms, like "running" to "run". This helps reduce unnecessary and redundant terms and ensures a faster runtime.

3.3 Implementation

Non-negative matrix factorization first requires that we have a matrix to factor, so we embed our corpus of lemmatized and cleaned words for each document into a matrix using term frequency - inverse document frequency (tfidf) using sklearn's implementation. The $maxdf$ ignores terms that appear in more than 95% of the documents, $mindf$ ignores terms that appear in less than 2 documents, and $maxfeatures$ considers the top 1000 features ordered by term frequency across the corpus. We then initialize an instance of NMF from sklearn where the number of components is the number of reduced topics from the dataset and return the decomposed matrices W and H. We extract the topic names from H by passing the highest weighted terms in the topic-term matrix (H) back through the tfidf vectorizer and list their topics `feature_names[i] for i in topic.argsort()[-10:]]`.

We go further to select an optimal number of topics using hyperparameter selection with a given model's coherence score as described in section 2.4. We create a gensim word2vec model based on the cleaned training corpus we list the topics from each model as described above and find the similarities for pairs of topic words `for pair in combinations( term_rankings[topic_index], 2 ): pair_scores.append( w2v_model.wv.similarity(pair[0], pair[1]) )` and average the similarities within and then across the documents. We then find the most coherent topic number and corresponding model iteratively within a range of possible topic numbers `topic_nums = np.arange(10, 30, 2)`.

A snippet of our code is shown below.

```
# Initialize TfidfVectorizer
vectorizer = TfidfVectorizer(max_df=0.95, min_df=2, max_features=1000)

# Fit and transform the processed abstracts into TF-IDF
tfidf = vectorizer.fit_transform(newsgroup_df['processed_documents'])
```

We then implement the NMF algorithm using sklearn. We return a W (document-topic matrix) and H (topic-term) matrix.

```
def do_nmf(tfidf, n_topics):
    # Specify the number of topics
    nmf_model = NMF(n_components=n_topics)
    W = nmf_model.fit_transform(tfidf)  # Document-topic matrix   (n_samples, n_components)
    H = nmf_model.components_          # Topic-term matrix       (n_components, n_features)
    return W, H, nmf_model
```

```
# we chose a subset of 10 topics from our 20newsgroup dataset
n_topics = 10
W, H, nmf_model = do_nmf(tfidf, n_topics)
```

```
# return top ten topics and the top ten words corresponding to each topic
feature_names = vectorizer.get_feature_names_out()
for index, topic in enumerate(H):
    print(f"Topic #{index}:")
    print(",".join([feature_names[i] for i in topic.argsort()[-10:]]))  # Top 10 words per topic
```

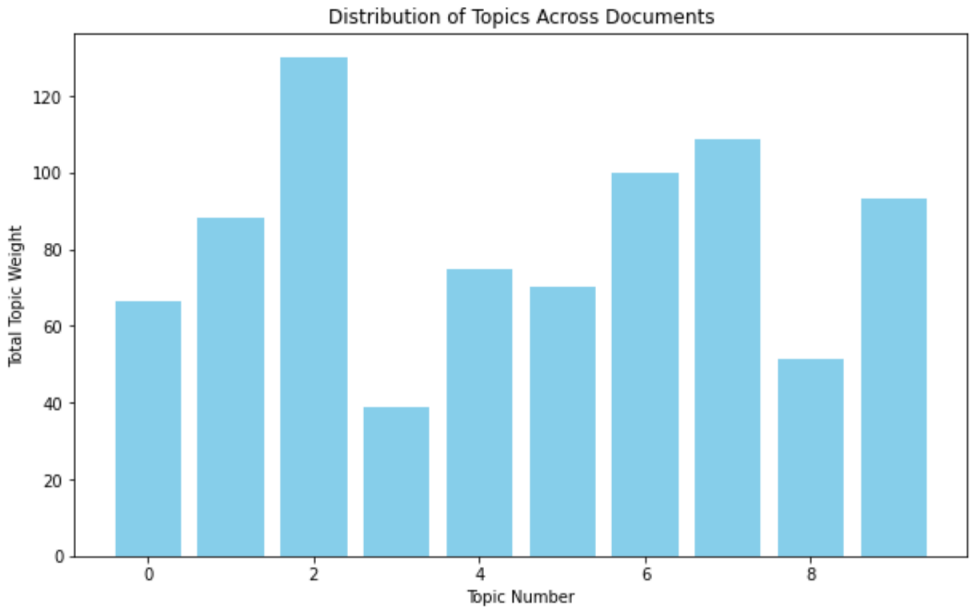
3.4 Results

These are the top ten words from each topic generated by NMF:

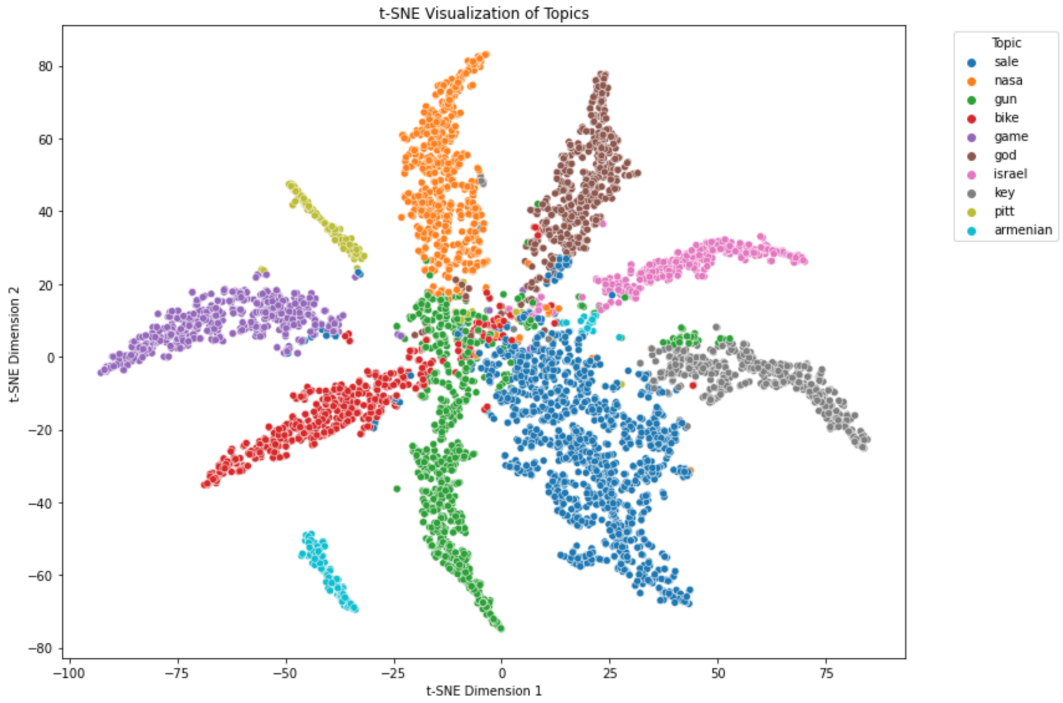
```
Topic #1:
say,believe,faith,bible,christ,people,church,jesus,christian,god
Topic #2:
crypto,phone,netcom,algorithm,government,escrow,encryption,clipper,chip,key
Topic #3:
software,new,distribution,offer,mail,host,posting,file,graphic,sale
Topic #4:
reply,computer,science,soon,univ,pittsburgh,bank,gordon,geb,pitt
Topic #5:
win,year,playoff,season,play,nhl,player,hockey,team,game
Topic #6:
state,right,peace,policy,jewish,palestinian,jew,arab,israeli,israel
Topic #7:
pat,alaska,jpl,moon,orbit,digex,access,gov,space,nasa
Topic #8:
handgun,crime,criminal,control,law,right,weapon,firearm,people,gun
Topic #9:
azeri,genocide,greek,turkey,serdar,argic,turk,armenia,turkish,armenian
Topic #10:
bnr,helmet,rider,riding,writes,dog,ride,motorcycle,dod,bike
```

We inspect the topics manually and find that they seem to resemble the 10 chosen newsgroups. For example, Topic 0 could be attributed as ""soc.religion.christian". The distribution of the top 10 words seem to be fairly consistent within each topic as well. For example, say, believe, faith, bible, christ, people, church, jesus, christian, god all relate to religion. Overall, our model was able to detect most of the 10 overall topics, with some difficulties for the graphic, sale, mideast and medicine topics.

We confirm our inspection visually with a barchart of number of documents per NMF topic. We find more evidence that NMF captured the original themes of the data as the values with the lowest topic weight are the topics that did not match our themes, i.e. topic 3 and topic 8.



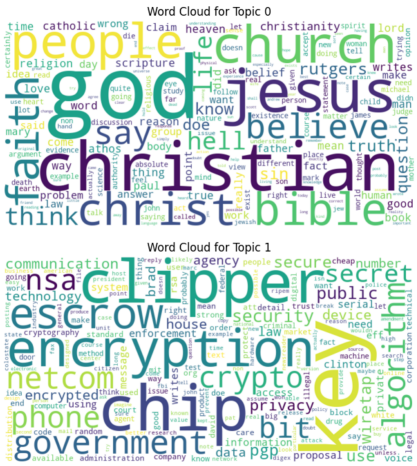
We take a look at how our model clustered topics using TSNE (t-Distributed Stochastic Neighbor Embedding), a statistical method for visualizing high-dimensional data by reducing it to lower-dimensional spaces. It provides a visual representation of how well-separated or cohesive our topics are.



Most of the topics form distinct clusters in the t-SNE plot, indicating that the topics are reasonably well-separated in the embedding space. "Game" (purple) and "God" (brown) form tight clusters, representing coherent groupings of documents. The sale cluster seems to be the biggest, which makes sense considering that it grouped sale and graphic together. Pitt appears to be very small, along with Armenian, suggesting that there aren't many documents that fall into these groups. There also seems to be a lot of overlap in the middle, which could result in incorrect clustering. The overlap between

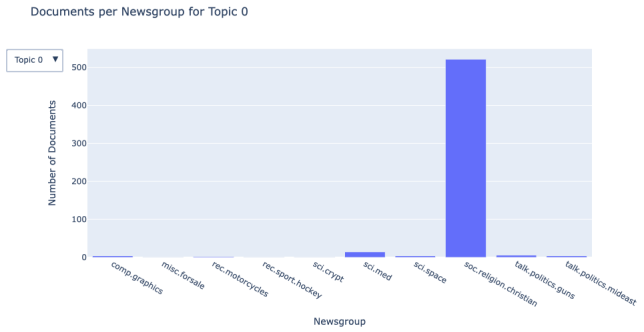
"Key" (gray), "Gun" (green), and "NASA" (orange) might reflect shared words or contexts in the documents. Topics like "Armenian" (cyan) and "Pitt" (yellow) are tightly packed, suggesting high coherence and similarity among documents in these topics. Topics like "Gun" (green) and "Sale" (blue) are more spread out, indicating that the documents assigned to these topics are more diverse and less well-defined. "NASA" (orange) and "God" (brown) are relatively close, which may reflect some contextual overlap, perhaps due to discussions about space and philosophy. A few points appear scattered outside the main clusters (e.g., lone points near the middle or periphery). These could represent noisy documents that do not fit well into any topic or documents with mixed themes that overlap multiple topics.

We also visualize the distribution of words for each topic using wordclouds.



Topic 0 appears to represent religion fairly well and topic 1 represents cryptography very well. The rest of the word clouds also show good representations for each topic, with topics 3 and 9 showing some irrelevant terms.

We can get the topic that is most relevant to a given document based on the W matrix that contains the probabilities of topics per document. We extract the most dominant topic based on the highest probability in order to map documents to topics.



Topic 0 (e.g., say, believe, bible, god) was dominant in soc.religion.christian.

Topic 1 (e.g., crypto, encryption, key) was dominant in sci.crypt.

Topic 2 (e.g., software, sale, file) appeared mostly in comp.graphics and misc.forsale.

Topic 3 (e.g., reply, science, pitt) was primarily in sci.med (140/500 documents).

Topic 4 (e.g., hockey, game, team) was dominant in rec.sport.hockey.

Topic 5 (e.g., israel, palestinian, peace) appeared in talk.politics.mideast (350/550 documents).

Topic 6 (e.g., nasa, moon, space) was seen in sci.space with some in sci.med.

Topic 7 (e.g., gun, handgun, crime) was mostly in talk.politics.guns, with some overlap in sci.med, sci.crypt, and talk.politics.mideast.

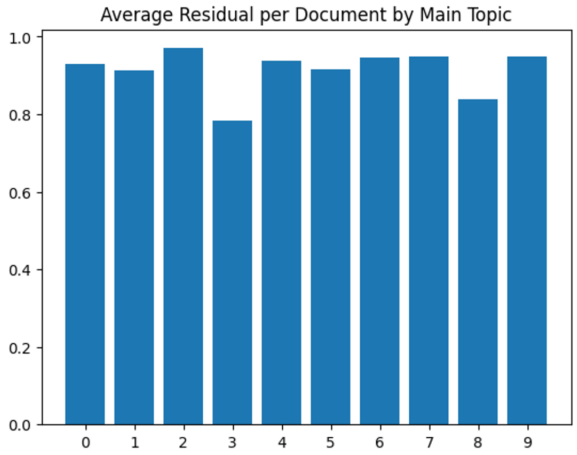
Topic 8 (e.g., armenian, turkey, genocide) was present in talk.politics.mideast (150/550 documents).

Topic 9 (e.g., bike, ride, motorcycle) dominated rec.motorcycles, with some in sci.med.

All topics fully covered 500 documents per newsgroup, except topics 3, 5, and 8.

We find that the topic diversity of the NMF implementation is **0.98**.

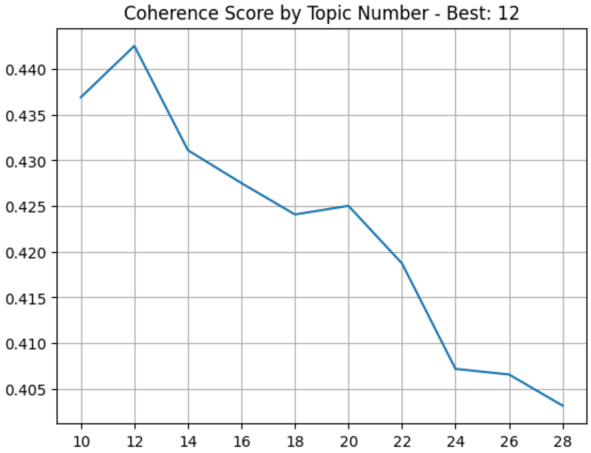
We find that the average residual score is **0.94**.
Broken down by topic, the residuals are nearly indistinguishable.



We inspect 20 random documents and compare them to their main topic and find that topics are representative of all of the the randomly chosen instances.

We find that the coherence score of the model is **0.43**.

When we use the coherence score to tune the number of topics as a hyperparameter. We find that the optimal number of topics is 12 and it results in a coherence score of **0.44**. This tuned mode has a residual of **0.93**, and is therefore very similar in performance to the model with a manually chosen topic number.



3.5 SOTA Implemenation

We implement bert using the implementation by Maarten Grootendorst. We restrict the number of topics to 10 to be able to digest them and compare better with NMF.

First we have to create custom embeddings to pass in to bert using `SentenceTransformer("all-MiniLM-L6-v2")` . These embeddings encode every document in the training set. Then we initialize a UMAP model to reduce the dimensionality of the embeddings `UMAP.UMAP(random_state=35)`, with a random state of 35 so we can obtain the same result with each run of the code. UMAP uses ideas from topological data analysis to find a simple representation for our embeddings. We also initialize a hierarchical clustering algorithm `HDBSCAN(min_cluster_size=50 )` to create topic groups and limit the topic size to 10. For more comparable results, we implemented the same vectorizer from NMF, `TfidfVectorizer(min_df=1, max_df=0.95, max_features=1000)` to extract each cluster's topics.

```
# Pre-calculate embeddings
data = newsgroup_df["processed_documents"]

embedding_model = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = embedding_model.encode(data, show_progress_bar=True)
```

```
#method for dimensionality reduction
umap_model = UMAP.UMAP(#n_neighbors=50,
                  #min_dist=0.5,
                  #metric='cosine',
                  random_state=35)
```

```
#higherarchial clustering method
hdbscan_model = HDBSCAN(min_cluster_size=50,
                        #min_samples=30
                       )
```

```
#vectorizer to create matrix from corpus
vectorizer_model = TfidfVectorizer(min_df=1, max_df=0.95, max_features=1000)
```

```
topic_model = BERTopic(

  # Pipeline models|
  embedding_model=embedding_model,
  umap_model=umap_model,
  hdbscan_model=hdbscan_model,
  vectorizer_model=vectorizer_model,

  # Hyperparameters
  top_n_words = 10,
  verbose=True
)

# Train model
topics, probs = topic_model.fit_transform(data, embeddings)
```

3.6 SOTA Results

Our BERTopic pipeline returns these ten topics:

```
1           [encryption, gun, clipper, firearm, escrow, chip, privacy, government, nsa, crypto]
2                      [bike, dod, motorcycle, bmw, rider, ride, behanna, helmet, riding, nec]
3                [nhl, hockey, team, playoff, player, season, puck, espn, lemieux, islander]
4           [jesus, bible, church, christ, faith, sin, christianity, scripture, athos, catholic]
5                      [geb, pitt, msg, dyer, patient, disease, food, candida, gordon, health]
6                  [sale, shipping, printer, disk, manual, forsale, floppy, ohio, item, brand]
7              [nasa, orbit, launch, satellite, spacecraft, shuttle, moon, mission, jpl, lunar]
8                      [image, jpeg, polygon, format, gif, vga, algorithm, pixel, window, mode]
9                 [israeli, israel, arab, jew, lebanese, palestinian, cpr, gaza, bony, hernlem]
10    [armenian, turkish, armenia, turk, azerbaijani, azerbaijan, turkey, argic, azeri, serdar]
```

Overall, our model was able to detect most of the 10 overall topics, with some difficulties for the gun, cryptography, and mideast. It was able to detect 8 out of 10 themes; gun and crypto were grouped and mideast was split into two, similar to NMF.

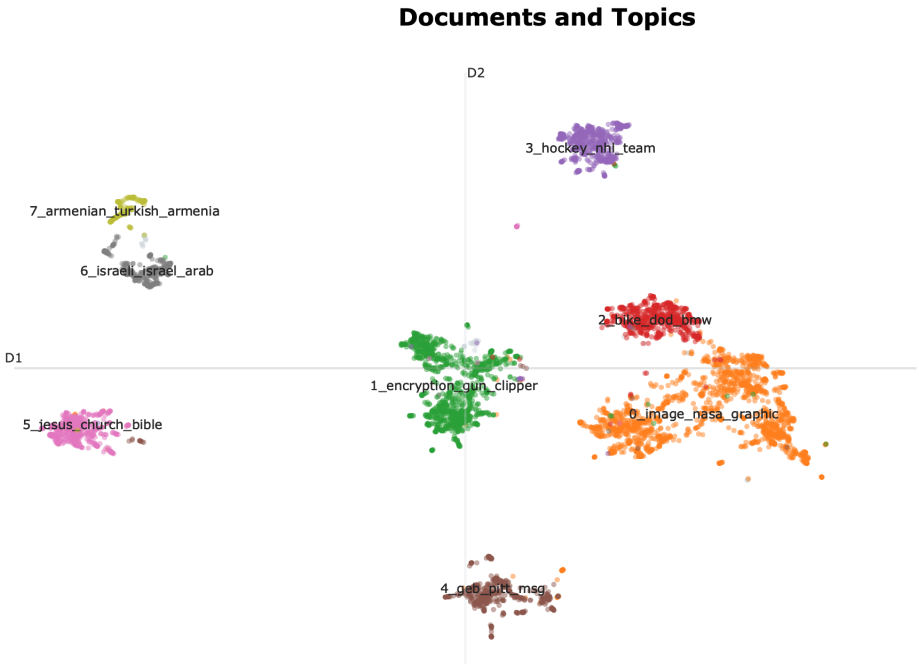
When we inspect each of the topics individually we find that most of the topics capture ideas that we know were contained within the original dataset, but some like "gebb_pit_msg" don't particularly

match topics we'd imagine from the set. If we dive deeper into the representation, its likely that this topic is capturing medical information with the words "disease" and "patient".

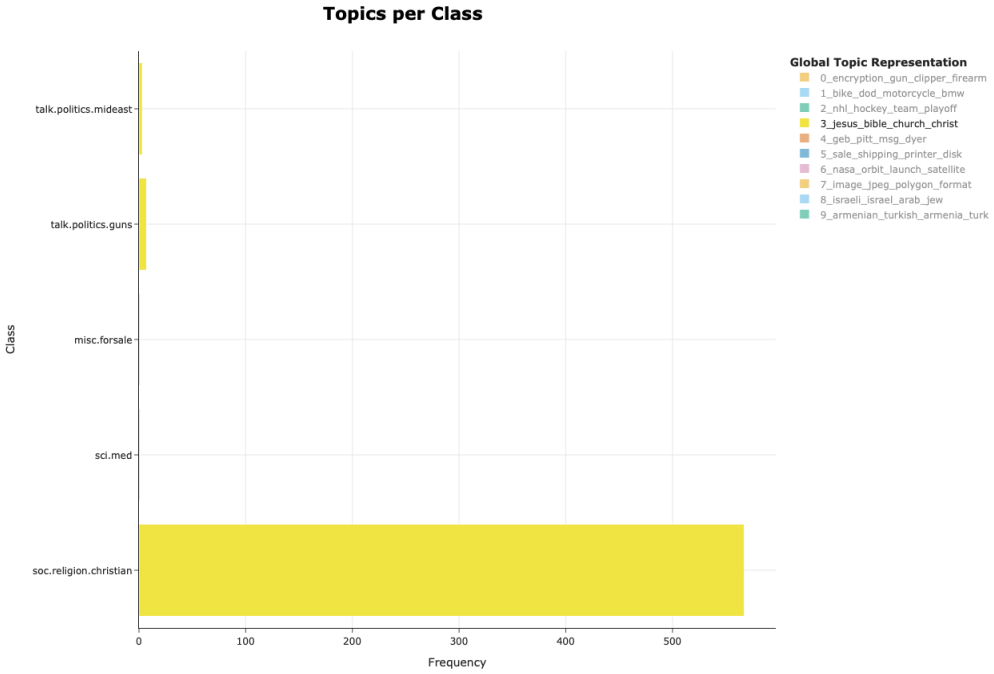
We also visualize these topics in wordclouds and find more convincing evidence that BERTopic captured the main themes.

When we go further an examine the distributions of the original documents labels within each topic, we find that except for topics 9 and 10, all of the other topics were able to get the full scope of the 500 documents in each newsgroup.

When we visualize the documents by their topic in 2d space, we confirm this finding. This visualization makes it clear that each of the topics are distinct across the original documents, with minimal overlap. It's reassuring that the middle-east related topics are close to each other in the reduced space.



We can look at the distribution of dominant topics per newsgroup for BERTopic as well.



Topic 1 (e.g., encryption, gun, clipper, privacy, government) was dominant in talk.politics.guns and sci.crypt.

Topic 2 (e.g., bike, motorcycle, helmet, ride) was mostly in rec.motorcycles, with some from misc.forsale.

Topic 3 (e.g., hockey, team, playoff, player) appeared primarily in comp.graphics and rec.sport.hockey.

Topic 4 (e.g., jesus, bible, church, faith) was dominant in soc.religion.christian.

Topic 5 (e.g., health, patient, disease) was from sci.med.

Topic 6 (e.g., sale, printer, shipping) was mostly from misc.forsale, with some in comp.graphics.

Topic 7 (e.g., nasa, satellite, mission, moon) was from sci.space.

Topic 8 (e.g., image, jpeg, gif, pixel) was dominant in comp.graphics.

Topic 9 (e.g., israeli, arab, palestinian) was from talk.politics.mideast (350/550 documents).

Topic 10 (e.g., armenian, turkey, azerbaijani) was from talk.politics.mideast (150/550 documents).

Except for topics 9 and 10, all other topics covered the full scope of 500 documents in each newsgroup.

We find that the coherence of the BERTopic model is **0.51**.

We find that the diversity of the BERTopic model is **0.99**

3.7 Compare and contrast

Our models seem to be performing very similarly. BERTopic seems to cluster the groups better, whereas the NMF model had a better representation of our original themes. Based on our visualizations, BERTopic was able to map topics to their corresponding newsgroup documents with more precision; NMF showed more variability in document distribution per topic. Both models incorrectly split the mideast group into two themes and both models grouped two themes into one (gun and crypto for bert and graphics and sale for NMF).

We find the same results when comparing the two models quantitatively.

Measure	NMF	BERTopic
Coherence	0.43	0.51
Diversity	0.98	0.99

BERTopic does incrementally better than NMF in terms of Coherence and Diversity. However, the themes in NMF noticeably captured the larger themes more coherently.

4 Conclusion

Overall, NMF appears to extract abstract themes, like religion and sports, better, whereas BERTopic appears to extract more specific topics, like christianity and hockey, better. BERTopic also produces more coherent top words across topics and groups the clusters better. BERTopic also required significantly longer runtime compared to NMF. Based on these pros and cons, the preference for BERTopic and NMF depends on the needs of a project. Our project favored the BERTopic model for its ability to better map documents to their original topics and for more consistent top words.

5 Acknowledgement

Thank you Dr. Tsui-Weng for teaching us the Linear Algebra we needed to succeed in this project. Thank you Somanshu Singla and Yashowardhan Shinde for being such outstanding TA's.

Thank you Maarten Grootendorst (Bert) for letting us use your algorithm.

References

[1] How hdbscan works¶. https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html.

[2] 20 newsgroups. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html, 1999. Last modified: 9 September 1999.

[3] What percentage of data is unstructured? 3 must-know statistics. https://edgedelta.com/company/blog/what-percentage-of-data-is-unstructured, 2024.

[4] A. Bhangale. Introduction to topic modelling with lda, nmf, top2vec and bertopic. https://medium.com/blend360/introduction-to-topic-modelling-with-lda-nmf-top2vec-and-bertopic-ffc3624d44e4, 2023.

[5] M. R. A. Both and A. Hinneberg. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, page 399–408, New York, NY, USA, 2015. Association for Computing Machinery.

[6] H. M. A. Subakti and N. Hariadi. The performance of bert as data representation of text clustering. *J Big Data*, 2022.

[7] Bindel. Numerics for data science, 2018-06-21. https://www.cs.cornell.edu/~bindel/class/sjtu-summer18/lec/2018-06-21.pdf, 2018.

[8] Blend360. Topic modelling: A comparison between lda, nmf, bertopic, and top2vec (part i). https://medium.com/blend360/topic-modelling-a-comparison-between-lda-nmf-bertopic-and-top2vec-part-i-3c16372d51f0#:~:text=Summary%20of%20Model%20Comparisons&text=While%20both%20BERTopic%20and%20NMF,deeper%20insights%20into%20the%20data., 2024. Accessed: 2024-12-09.

[9] C. Goyal. Part 15: Step by step guide to master nlp – topic modelling using nmf. https://www.analyticsvidhya.com/blog/2021/06/part-15-step-by-step-guide-to-master-nlp-topic-modelling-using-nmf/, 2024.

[10] R. Egger and J. Yu. A topic modeling comparison between lda, nmf, top2vec, and bertopic to demystify twitter posts. *Frontiers in Sociology*, 7, 05 2022.

[11] F. Chiusano. Two minutes nlp — learn tf-idf with easy examples. https://medium.com/nlplanet/two-minutes-nlp-learn-tf-idf-with-easy-examples-7c15957b4cb3#:~:text=Term%20Frequency%20(TF)%3A%20how,common%20words%20have%20low%20scores., 2022.

[12] FreeCodeCamp. How to process textual data using tf-idf in python. 2024. Accessed: 2024-12-02.

[13] Geeks for Geeks. Non-negative matrix factorization. https://www.geeksforgeeks.org/non-negative-matrix-factorization/, 2023.

[14] Geeks for Geeks. Topic modeling examples. https://www.geeksforgeeks.org/topic-modeling-examples/, 2024.

[15] GeeksforGeeks. Python word embedding using word2vec. /urlhttps://www.geeksforgeeks.org/python-word-embedding-using-word2vec/, n.d. Accessed: 2024-12-04.

[16] M. Grootendorst. Bertopic: Visualization, 2024. Accessed: 2024-12-04.

[17] ProjectPro. A beginner's guide to topic modeling nlp. https://www.projectpro.io/article/topic-modeling-nlp/801, 2024.

[18] Qualtrics. Topic modeling: definition, benefits and use cases. https://www.qualtrics.com/experience-management/research/topic-modeling/.

[19] S. Kim. Let us extract some topics from text data — part iv: Bertopic. https://towardsdatascience.com/let-us-extract-some-topics-from-text-data-part-iv-bertopic-46ddf3c91622, 2022.

[20] ScienceDirect. Frobenius norm. https://www.sciencedirect.com/topics/engineering/frobenius-norm, n.d. Accessed: 2024-12-04.

[21] J. Silge. Evaluating topic models. https://juliasilge.com/blog/evaluating-stm/, 2020. Accessed: 2024-12-04.

[22] B. Slawski. Semantic topic modeling for search queries at google. https://gofishdigital.com/blog/semantic-topic-modeling/, 2016.

[23] SOAX. How much data is generated every day? https://soax.com/research/data-generated-per-day, 2024.

[24] N. K. Tran, S. Zerr, K. Bischoff, C. Niederee, and R. Krestel. Topic cropping: Leveraging latent topics for the analysis of small corpora. volume 8092, pages 297–308, 09 2013.

[25] H. Vijay. Dimensionality reduction : Pca, tsne, umap. https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/, May 2023.