Mesgari, Nastaran

nmesgari@constructor.university

Analysis of Repetition in Teaching

Contents

[1 Executive Summery 2](#_Toc191638755)

[2 Introduction 3](#_Toc191638756)

[3 Methods of Preprocessing and finding Similarity 5](#_Toc191638757)

[3-1 TF-IDF (Term Frequency – Inverse Document Frequency) 6](#_Toc191638758)

[3-2 Latent Dirichlet Allocation (LDA) 7](#_Toc191638759)

[3-3 Process and Results of Similarity Evaluation 7](#_Toc191638760)

[3-3-1 Preprocessing: Basic Cleaning and Standardization 8](#_Toc191638761)

[3-3-2 Text Refinement: Tokenization and Bigram Extraction 9](#_Toc191638762)

[4.1.1 summarize: 9](#_Toc191638763)

[4.1.2 Reasons for High Similarities: 10](#_Toc191638764)

[4.2 Latent Dirichlet Allocation (LDA) 10](#_Toc191638765)

[4.2.1 Methodology 10](#_Toc191638766)

[4.5 Comparison Between LDA and Cosine Similarity Calculation 11](#_Toc191638767)

[4.5.1 Key Differences: 11](#_Toc191638768)

[4.5.2 Comparison of Results: 12](#_Toc191638769)

[4.5.3 Why the LDA Was Used After the Cosine Similarity Calculation 13](#_Toc191638770)

[4.6 Use Named Entity Recognition (NER): 13](#_Toc191638771)

[4.6.1 Key Features of the Code: 13](#_Toc191638772)

[4.6.2 Expected Results: 14](#_Toc191638773)

[4.15 Why Was This Update Introduced? 15](#_Toc191638774)

[4.16 Key Features of the Code: 15](#_Toc191638775)

[4.17 Function Breakdown: 16](#_Toc191638776)

[4.18 Expected Output: 16](#_Toc191638777)

[4.19 Why This Code is Useful: 16](#_Toc191638778)

[4.20 Potential Improvements/Modifications: 16](#_Toc191638779)

[4.21 Potential Improvements: 17](#_Toc191638780)

[4.22 Key Steps Taken: 19](#_Toc191638781)

[4.23 Results: 20](#_Toc191638782)

[4.24 How I Reached This: 20](#_Toc191638783)

[4.25 Explanation of the Code: 20](#_Toc191638784)

[4.26 How Similarity is Calculated: 21](#_Toc191638785)

[4.27 Result: 21](#_Toc191638786)

[4.28 How This Result is Reached: 21](#_Toc191638787)

[5 Conclusion: 21](#_Toc191638788)

[6 Suggestions for Next Steps: 22](#_Toc191638789)

[6.1 Summary: 22](#_Toc191638790)

# 1 Executive Summery

To find duplications and raise the standard of the curriculum overall, this project focuses on comparing the content of two courses: Artificial Intelligence in Business & Society and Visual Communication and Data Storytelling (*for simplicity, the course "Visual Communication and Data Storytelling" will be referred to as "Visualization," and "Artificial Intelligence in Business & Society" will be referred to as "AI" throughout the text*). Two courses were chosen: one on AI, which covered subjects like data models, algorithms, and machine learning," and the other on visualization, which emphasized graphical data representation, storytelling, and insight generation. To guarantee accurate analysis, the course content was prepared using several natural language processing (NLP) techniques. By removing stop words, digits, and special characters and applying lemmatization, the text becomes standardized, reducing noise and improving the accuracy of downstream models, and I completed my preprocessing.

After preprocessing, domain-specific keywords were identified using SEO techniques to discover related concepts on various websites*.* Depending on the reconstructed results of the course material and my understanding *(during the finding similarity, I checked out the keywords and changed them to find the best ones.)*, I selected the most relevant keywords for each subject.   
  
• AI focuses more on technology, algorithms, and models, with keywords like “ai”, “learning”, “machine”, “intelligence”, and “algorithm”.

• Visualization emphasizes conveying information and storytelling through data, with keywords such as “message”, “visualization”, “context”, “story”, “audience”, “insights”, and “experience”.

I tried to find similarities by using techniques of NLP, but there was a very high similarity. After cleaning the data, I found a minimum of about 47% similarity. This percentage is not acceptable to me because I do not think we have this similarity between these courses, so I decided to use another way to find the similarity. In this way, I calculated the similarity based on the frequency of keywords. Based on my keywords: **82 AI-related keywords and 27 visualization-related keywords** were found in the AI course, according to a keyword frequency analysis. There were **216 keywords related to visualization and 123 related to AI in the visualization course**. similarity score of **0.2433** was obtained from this overlaps, suggesting a moderate degree of redundancy between the two courses. To further show the distribution of keywords and give a visual depiction of the most common terms in the materials, word clouds were created for both courses.

The results indicate that although the courses retain their distinct, there are significant conceptual overlaps, especially in fields pertaining to data and model application. This overlap suggests possible duplications that, if resolved, could improve the uniqueness and efficacy of each course. In order to find more extensive redundancy patterns throughout the curriculum, the analysis could be extended to include more courses as a subsequent step. To learn more about content overlaps, advanced Natural Language Processing (NLP) methods like topic modeling and similarity analysis could be used. These results could guide focused content optimization tacti, guaranteeing a distinct distinction between courses and enhancing students’ overall educational experience. To improve course content, reduce duplication, and develop a more coherent curriculum, cooperation with instructors is advised.

This project lays the groundwork for future initiatives to maximize academic programs by highlighting the significance of assessing instructional materials for alignment and redundancy. University can improve learning outcomes, encourage innovation, and preserve student learning by making sure that courses are streamlined and complementary. The insights gained from this analysis are instrumental in achieving these goals and improving the quality of teaching and learning.

# 2 Introduction

In academic settings, the quality and structure of course content significantly impact students’ learning experiences and outcomes. With the growing importance of fields like Artificial Intelligence (AI) and Data Visualization, the need for well-structured and non-redundant educational programs has become increasingly critical. Redundant content, while sometimes unavoidable, can lead to inefficiencies in learning, reduce engagement, and limit students’ exposure to new concepts. Addressing such overlaps is essential for optimizing course offerings and ensuring students receive a comprehensive yet streamlined education.

This project aims to analyze and evaluate the extent of content similarities between two courses, one focusing on AI and the other on Visualization. These domains, while distinct, often intersect in their discussion of data-driven decision-making and the application of models to real-world problems. By systematically identifying overlaps, the project seeks to highlight areas of potential redundancy and offer solutions for improving course alignment.

The analysis is centered on identifying keywords representative of the core themes in each course. Preprocessing techniques are applied to refine the course materials for accurate assessment, followed by similarity metrics to quantify the degree of overlap. The results are visualized through tools such as Word Clouds, providing a clear picture of the shared and unique elements of the courses.

This initiative not only supports the academic goals of content optimization but also fosters collaboration among instructors. By aligning course objectives and minimizing redundancies, educators can offer more diverse and impactful learning experiences. This project serves as a steppingstone for broader efforts to evaluate and enhance curriculum design, ultimately contributing to higher standards in education and better preparing students for the challenges of an ever-evolving professional landscape.

To carry out this project, a step-by-step approach was followed to properly analyze the content similarity between two courses related to “Artificial Intelligence” and “Data Visualization.” These steps included data identification, text processing, feature extraction, and similarity analysis. Below is a breakdown of each step:

**1. Course Selection** Initially, two courses related to “AI” and “Visualization” were chosen as the subjects for the similarity analysis. The course materials were sourced from transcripts of lecture recordings, course slides, and supplementary reading materials provided as part of the curriculum.

For each course, the corresponding text files were extracted and prepared for processing:

**Course 1: AI2.docx:** This file contains content related to AI, including lecture transcripts, instructor notes, and key discussion points from 2 recorded sessions, each approximately 60–90 minutes long.

**Course 2: Vir1.docx:** This file consists of materials from the Data Visualization course, including slides, textual descriptions, and transcripts from 2 recorded sessions, with an average length of 60–90 minutes per session.

**2. Definition of Key Terms** Next, a set of key terms related to the topics of each course was defined. These terms were used to identify and compare the similarities between the courses:

For the AI course: {ai’, ‘machine’, ‘learning’, ‘intelligence’, ‘algorithm’, ‘data’, ‘model’}   
For the Visualization course: {‘story’, ‘narrative’, ‘data’, ‘visualization’, ‘insights’, ‘message’, ‘emotion’}

**3. Text Processing and Cleaning** The content of both text files was processed to remove any noise and irrelevant data. This step involved converting the text into a standard format, making it ready for analysis to ensure the results were accurate.

**4. Keyword Analysis** At this stage, the number of key terms found in each course was counted. This count served as a metric for comparing the similarities and differences:

In AI: 82 AI-related terms and 27 Visualization-related terms.

In Visualization: 123 AI-related terms and 216 Visualization-related terms.

**Similarity Formula Explanation:**

In this project, a simple formula is used to calculate the similarity between two courses, one related to "Artificial Intelligence" (AI) and the other to " Visualization." This formula computes similarity based on the number of shared keywords between the two courses. Specifically, it compares the keywords associated with each topic (AI and Visualization) in the course materials.

**Steps for Calculating Similarity:**

1. **Counting Keywords**: First, the number of keywords related to AI and Visualization is counted in both courses. These keywords are specific terms directly related to each topic. For example, for the "Artificial Intelligence" course, the keywords include terms like "AI," "machine learning," "data," and "algorithm." For the "Visualization" course, keywords such as "story," "narrative," "insights," "visualization," and "audience" are considered.
2. **Calculating Shared Keywords**: Next, the number of shared keywords between the two courses is calculated for each topic (AI and Visualization). The minimum count of the shared keywords for each topic is taken between the two courses.
3. **Similarity Formula**: The similarity is calculated using the following formula:

​

Where:

* are the counts of AI-related keywords in the two courses.
* are the counts of Data Visualization-related keywords in the two courses.
* represent the minimum counts of shared keywords for each topic (AI and Visualization) between the two courses.

This formula calculates the similarity based on the shared keywords for each topic, and the sum of these similarities is then divided by the total number of keywords for both courses.

**Conclusion:**

This formula provides an initial metric to assess the similarity between two courses. The resulting similarity score is a value between 0 and 1, indicating the degree of similarity or conceptual overlap between the two courses. A value closer to 1 suggests a high degree of similarity, while a value closer to 0 indicates significant differences.

# 3 Methods of Preprocessing and finding Similarity

In this project, I employed various stages of preprocessing and similarity computation to explore the relationship between documents. During the preprocessing phase, I applied different techniques to clean and prepare the text for further analysis, aiming to enhance the quality and accuracy of similarity measurement.

Initially, I performed basic text cleaning tasks such as converting the text to lowercase, removing punctuation, and eliminating irrelevant numbers. After that, I tokenized the text to split it into individual words and used n-grams (including bigrams) to capture multi-word patterns, enhancing the representation of the document.

In subsequent steps, I calculated the similarity between documents in different ways. First, I calculated simple similarity based on word overlap and frequency, and then I moved to more advanced methods, including **TF-IDF (Term Frequency-Inverse Document Frequency)** and **LDA (Latent Dirichlet Allocation)**, to model the textual data and extract meaningful features for comparison.

By adjusting the preprocessing techniques and similarity measures at each stage, I compared the results and analyzed how these changes affected the final similarity values. Through this iterative process, I aimed to provide a comprehensive understanding of the methods and their impact on similarity assessment

## 3-1 TF-IDF (Term Frequency – Inverse Document Frequency)

**TF-IDF (Term Frequency - Inverse Document Frequency)** is a statistical measure used in text mining and information retrieval to assess the importance of a word within a document relative to a collection or corpus of documents.

**Explanation of TF-IDF:**

1. **Term Frequency (TF)**:
   * **TF** measures how frequently a term (word) appears in a document.
   * It is calculated by the formula:
   * The more often a word appears in a document, the higher its term frequency. This helps capture the relevance of a word within that document.
2. **Inverse Document Frequency (IDF)**:
   * **IDF** measures the importance of a word across all documents in the corpus.
   * It is calculated by the formula:

* + Words that appear in many documents (e.g., common words like "the", “and” "is") will have a low IDF score, meaning they are less informative.
  + Words that appear in fewer documents will have a higher IDF score, indicating that they are more specific to certain documents.

**The TF-IDF Score:**

The **TF-IDF score** is the product of **TF** and **IDF**:

TF-IDF(t)=TF(t)×IDF(t)

* A high **TF-IDF** score indicates that a word is important in the given document but not common across all documents, making it more relevant or specific.

**Intuition Behind TF-IDF:**

* **TF** tells you how relevant a word is within a specific document.
* **IDF** tells you how rare or common a word is in the entire corpus.
* **TF-IDF** combines these two to identify words that are both frequent within a document but rare across all documents.

**TF-IDF** is an effective method for identifying important words in text documents by considering both their frequency within a specific document and their rarity across a corpus. This balance helps to highlight terms that contribute to the unique meaning of a document, making it a crucial tool in text analysis.

text processing and cleaning involved extracting content from .docx files, removing unnecessary numerical data from filenames, and preparing the content by structuring it into a usable format for analysis. This process is vital to ensure that only the relevant text data is considered in subsequent steps of the analysis.

## 3-2 Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model used in natural language processing (NLP) and machine learning to discover the underlying topics in a collection of documents. LDA assumes that each document is a mixture of a small number of topics and that each word in the document is attributable to one of the document's topics. Here's a brief breakdown of how it works:

1. **Topic Modeling**: LDA aims to uncover hidden thematic structures in a large set of text data. It represents documents as a mixture of various topics, where each topic is a distribution over words.
2. **Document Representation**: Each document is considered a mixture of multiple topics. The assumption is that each document is generated by selecting a topic distribution and then selecting words based on that topic.
3. **Inference**: LDA tries to infer the topic distribution for each document and the word distribution for each topic, based on the observed words in the documents.

## 3-3 Process and Results of Similarity Evaluation

In this section, we detail the steps taken to evaluate the similarity between documents using different methods. The goal is to assess how well these methods capture the semantic similarity between text documents. We started with basic text preprocessing, moved through various techniques like tokenization and n-grams extraction, and then used different models such as TF-IDF and Latent Dirichlet Allocation (LDA) to measure similarity. After each step, we present the results and compare them to gain insights into how each method contributes to the overall similarity evaluation.

### 3-3-1 Preprocessing: Basic Cleaning and Standardization

In this step, the text was preprocessed by converting it to lowercase, removing numbers, and eliminating punctuation marks. This basic cleaning step aimed to standardize the text and remove any unnecessary elements that could interfere with the analysis. In the next phase, **tokenization** and **bigram extraction** were added, which will expand the textual representation by breaking down the text into smaller units.

1. First Step: after loading the library, I read the text of each document and after that for preprocessing and cleaning the data I use these methods include the following steps:

* **Lowercasing:**  
  First, all texts were converted to lowercase to eliminate any potential discrepancies between uppercase and lowercase letters
* **Number Removal:**  
  Next, all numbers were removed from the text. Numbers are generally not useful for semantic analysis and can reduce the accuracy of the analysis, so removing them is necessary.
* **Punctuation Removal:**  
  After that, all punctuation marks like periods, commas, and question marks were removed from the text. Punctuation can introduce noise into the analysis and does not contribute to the meaning of the text.

1. Second Step: by using TF-IDF (Term Frequency - Inverse Document Frequency) Converting Text to Numerical Vectors.
2. Third Step: for finding Similarity by using Cosine Similarity Calculation

**Cosine Similarity Calculation**

Once the text was transformed into TF-IDF vectors, **Cosine Similarity** was used to quantify the similarity between two documents. The formula used for cosine similarity is:

Where:

A and B are the **TF-IDF vectors** of the two documents.

represents the **dot product** of the two vectors.

and are the **magnitudes (norms)** of the vectors.

The result is a similarity score between **0 and 1**, where a value closer to **1** indicates higher similarity.

***result TF-IDF: Similarity between AI2.docx and Vir1.docx: 0.8910***

***result LDA: Similarity between AI2.docx and Vir1.docx 0.4458***

This result shows high similarity between these courses. This result is not acceptable to me because it shows we high result, so I decided to improve that by using another method for cleaning and then finding the similarity between the two documents.

### 3-3-2 Text Refinement: Tokenization and Bigram Extraction

To enhance the accuracy and logical reasoning in measuring document similarity, I applied several Natural Language Processing (NLP) techniques. First, I performed **text preprocessing**, which involved cleaning and preparing the text by removing unnecessary elements such as punctuation, numbers, and converting the text to lowercase. This step helps standardize the text, ensuring that irrelevant details don't interfere with the analysis and makes the text cleaner and easier to process.

Next phase after Basic Cleaning and Standardization, I applied **tokenization**, which is the process of splitting the text into smaller units, typically words or phrases, known as tokens. Tokenization allows for the examination of individual words, making it a crucial step for tasks like frequency analysis or comparison, as it breaks down the text into manageable units for further processing.

Following that, I performed a **word frequency count**, where I counted the occurrences of each word in the document. Understanding word frequency is essential in text analysis because it helps determine which words are most significant in the context of a document, allowing for a more effective similarity calculation.

Additionally, I compared **common words** between documents. By identifying words that appear in both texts, I was able to measure the overlap between them, which serves as a key indicator of thematic similarity. Common words often reflect shared topics or themes, contributing to a more meaningful comparison.

Lastly, I extracted and compared **bigrams**, which are pairs of consecutive words. Bigrams capture more context than individual words, as they account for word pairs that frequently appear together, offering deeper insight into the relationships between words. This allowed me to improve the accuracy of the similarity calculation by incorporating context beyond individual word matches.

Using these methods, along with adjustments in preprocessing and the inclusion of bigrams, the similarity measure between the documents improved significantly. By applying **TF-IDF** and **Cosine Similarity** to the processed text, the calculated similarity between the documents reached ***49%***. This improvement highlights the effectiveness of the preprocessing steps and the addition of bigrams in capturing the deeper semantic relationship between the texts.

By applying for **Latent Dirichlet Allocation (LDA)** with an increased number of topics, I was able to improve the similarity measure between the two documents. The cosine similarity between **'AI2.docx' and 'Vir1.docx' was calculated to be *40%*,** showing a reasonable degree of similarity after adjusting the number of topics.

### ****4.1.1 summarize:****

The code) **Cosine Similarity Calculation (**is designed to systematically analyze the similarity between two text documents by comparing their content and structure. By applying preprocessing techniques, tokenization, and frequency analysis, it identifies shared terms, common bigrams, and overall thematic overlap. The results help assess the extent of redundancy between the documents, providing insights that can be used to refine course materials and ensure clearer content differentiation. But in this step, I do not recommend because based on this technique we have too redundancy. (0.8910)

### 4.1.2 ****Reasons for High Similarities:****

If there are many common words or bigrams between the documents, it may be due to the following reasons: 1. **Similar Topics:** Both documents might cover the same subject matter. 2. **Repetitive Phrases:** Standardized phrases or templates may have been used in both files. 3. **Insufficient Preprocessing:** Non-essential words (e.g., “the,” “is”) were not removed, affecting the similarity results.

Code

## 4.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) was applied as a topic modeling technique to analyze the thematic structure of the course materials. LDA is a probabilistic generative model that identifies hidden topics within a collection of documents by assuming that each document is composed of multiple topics, and each topic consists of a distribution of words.

### 4.2.1 Methodology

To implement LDA, the following steps were taken:

Preprocessing the Text: The course content was cleaned by removing stop words, converting text to lowercase, and tokenizing the words.

Transforming Text into a Bag-of-Words Model: The documents were represented numerically by converting words into frequency-based vectors.

Applying LDA: The model was trained to extract a predefined number of topics, each characterized by a set of words with different probabilities.

Interpreting Topics: The dominant topics within each course were identified by analyzing the most frequent words within each topic distribution.

Results and Interpretation

By applying LDA, we extracted key topics from both courses and analyzed their overlap. This helped in identifying conceptual similarities, redundancies, and distinctions between the two subjects. The insights gained from this analysis provide a deeper understanding of the thematic structure of the course materials, allowing for potential refinement in curriculum design.

LDA was particularly useful in quantifying content overlap, as it allowed us to move beyond simple keyword matching and explore deeper semantic relationships between topics in both courses. This approach contributes to a more systematic evaluation of course alignment and potential areas for differentiation.

**Why Use LDA:**

* The previous steps analyzed **word-level overlap** (e.g., shared words and phrases). However, they did not provide a high-level understanding of the themes or topics.
* LDA adds a layer of abstraction by grouping related words into topics, giving a clearer picture of the documents’ **semantic content**.
* This approach ensures we’re not just comparing superficial textual similarities but also deeper thematic alignments, which is crucial for making informed decisions in downstream tasks.

This code helps identify the main topics in the documents, providing a thematic overview that can be used to refine the preprocessing pipeline and guide further analyses. but as we can see the result is not good and we decided change the code and check the other ways.

Topic 1: [‘yeah’, ‘ok’, ‘just’, ‘think’, ‘right’, ‘data’, ‘like’, ‘maybe’, ‘bit’, ‘umm’]   
Topic 2: [‘that’, ‘ai’, ‘things’, ‘probably’, ‘say’, ‘yeah’, ‘decision’, ‘like’, ‘maybe’, ‘ok’]

Code

## 4.5 ****Comparison Between LDA and Cosine Similarity Calculation****

Both codes aim to extract **topics** from .docx files using **Latent Dirichlet Allocation (LDA)**, but there are significant differences in their **preprocessing steps** and the resulting outputs. For naming the code **Cosine Similarity Calculation- Cosine Code and for LDA: LDA Code,**

### 4.5.1 ****Key Differences:****

1. **Custom Stop words:**
   * **Cosine Code:** Relied only on the built-in English stop word list provided by Count Vectorizer.
   * **LDA Code:** Adds a **custom stop word list** (['yeah', 'ok', 'umm', 'just', 'like', 'bit', 'maybe', 'right', 'that’s']) to filter out common filler words that do not contribute to meaningful topics.
2. **Preprocessing:**
   * **Cosine Code:**
     + Lowercased text.
     + Removed digits and punctuation.
   * **LDA Code:**
     + Includes all previous preprocessing steps.
     + Additionally, it removes **custom filler words**, making the data cleaner and more topic focused.
3. **Output Topics:**
   * **Cosine Code:**
     + **Topic 1:** [yeah’, ok’, just’, ‘think’, ‘right’, ‘data’, ‘like’, ‘maybe’, ‘bit’, ‘umm’]
     + **Topic 2:** [that’s’, ai’, things’, ‘probably’, ‘say’, ‘yeah’, ‘decision’, ‘like’, ‘maybe’, ‘ok’]
     + These topics contain numerous filler words (e.g., “yeah,” “ok,” “maybe”) that obscure the main themes.
   * **LDA Code:**
     + **Topic 1:** [‘ai’, ‘things’, ‘probably’, ‘say’, ‘decision’, ‘kind’, ‘human’, ‘actually’, ‘basically’, ‘different’]
     + **Topic 2:** [‘think’, ‘data’, ‘lets’, ‘know’, ‘say’, ‘income’, ‘don’t’, ‘course’, ‘really’, ‘want’]
     + The removal of filler words results in more **coherent topics**, highlighting terms related to artificial intelligence, decision-making, and data.
4. **Clarity of Topics:**
   * The **LDA Code** generates more focused topics, making them easier to interpret and align with the context of the documents.

### 4.5.2 Comparison of Results:

**Cosine Similarity Approach**

**The cosine similarity method, while effective in quantifying overall textual similarity, had some limitations:**

**Presence of irrelevant filler words: Since the approach relies on raw text processing, it included many general-purpose words that did not contribute to meaningful topic differentiation. These words created noise in the similarity calculation, potentially inflating similarity scores.**

**Less distinct topic differentiation: Due to the presence of unfiltered words and lack of contextual understanding, the cosine similarity approach struggled to clearly separate different themes within the texts. As a result, the output primarily indicated general overlap without providing deeper insight into specific conceptual relationships.**

**Latent Dirichlet Allocation (LDA) Approach**

**The LDA model provided a more structured and insightful representation of the course materials:**

**By identifying key themes within the texts, LDA allowed for a clearer separation of subject areas.**

***Topic 1* focused primarily on artificial intelligence concepts, with terms such as “AI,” “decision,” and “different.” This suggests that one section of the course content emphasizes AI-related decision-making processes and the distinctions between AI methodologies.**

***Topic 2* was centered around data analysis, containing terms like “data,” “income,” and “course.” This indicates a focus on how data is analyzed in different contexts, possibly within business applications.**

***Improved topic clarity through stop word removal:* The preprocessing stage involved removing common and custom stop words that were not relevant to the subject matter. This significantly enhanced the quality of the extracted topics by reducing noise and ensuring that only meaningful terms contributed to topic formation.**

**Overall, the comparison between cosine similarity and LDA highlights the advantages of topic modeling in providing a more granular and interpretable analysis of course content. While cosine similarity offers a numerical similarity measure, LDA delivers thematic insights that help in better understanding and distinguishing conceptual overlaps. (*But in the result I found that I think Cosine Similarity Approach may be better than LDA because is not understandable for me 89% similarity. and the base on the theoretical reason we should assume LDA is better than Cosine Similarity)***

### 4.5.3 ****Why the LDA Was Used After the Cosine Similarity Calculation****

1. **Improvement in Preprocessing:**
   * Observing the prevalence of filler words in the previous output indicated a need for additional filtering. Adding a **custom stop word list** addressed this issue.
2. **Enhancing Interpretability:**
   * By focusing on meaningful words, the current approach provides **clearer and more actionable insights** into the themes in the documents.
3. **Aligning Topics with Goals:**
   * The improved topics can be better leveraged for tasks like document comparison, classification, or deeper semantic analysis.

## 4.6 ****Use**** Named Entity Recognition (NER)****:****

We want to improve the result of LDA by using an advanced preprocessing step, incorporating Named Entity Recognition (NER) using SpaCy. This enhances the clarity of topics extracted by the Latent Dirichlet Allocation (LDA) model.

### 4.6.1 ****Key Features of the Code:****

1. **Named Entity Recognition (NER) Filtering:**
   * The code uses SpaCy’s en\_core\_web\_sm model to identify and remove named entities (e.g., names, places, dates, organizations) from the text.
   * By excluding these entities, the focus shifts to general terms and patterns, reducing noise caused by specific names or details.
2. **Stop word Customization:**
   * The code expands the custom stop word list to include more filler words such as 'say', 'lets', 'think', 'know', 'course', 'really', 'probably'.
   * This refinement ensures that the resulting topics are not dominated by irrelevant or generic terms.
3. **Enhanced Preprocessing Pipeline:**
   * Convert text to lowercase.
   * Removes digits and punctuation.
   * Filters out named entities and stop words.
4. **Topic Modeling:**
   * The vectorization step (Count Vectorizer) converts the cleaned text into a **document-term matrix**.
   * **Latent Dirichlet Allocation (LDA)** is applied to extract two topics (n\_components=2).
   * The top 10 terms for each topic are displayed.

**Comparison with LDA without using NER:**

1. **Addition of NER:**
   * In previous versions, named entities (like “AI,” “human,” “decision”) were included in the topic extraction process. While these entities might occasionally be relevant, their overrepresentation could obscure broader patterns.
   * The current code removes such entities to focus on more general and **conceptual terms**.
2. **Improved Topic Clarity:**
   * By removing named entities, the extracted topics emphasize broader patterns instead of being skewed by document-specific details.
3. **Expanded Stop words List:**
   * The expanded stop word list further reduces noise and highlights meaningful words.

### ****4.6.2 Expected Results:****

* **Topics from LDA without NER:**
  + **Topic 1:** [‘ai’, ‘things’, ‘probably’, ‘say’, ‘decision’, ‘kind’, ‘human’, ‘actually’, ‘basically’, ‘different’]
  + **Topic 2:** [‘think’, ‘data’, ‘lets’, ‘know’, ‘say’, ‘income’, ‘dont’, ‘course’, ‘really’, ‘want’]
* **Topics from LDA with NER:**
  + The new topics are expected to:
    - Exclude specific names and entities.
    - Highlight key themes or terms related to the general context of the documents.
    - Example: [“decision-making,” “data analysis,” “technological impact”].

**Advantages of This Code for Future Steps:**

1. **Generalization:**
   * Topics derived from this process are likely to generalize better across different datasets since named entities and filler words are removed.
2. **Suitability for Downstream Tasks:**
   * The output can be used for:
     + **Document classification or clustering.**
     + **Thematic comparison** across files.
     + **Keyword extraction** for summarization.
3. **Focused Analysis:**
   * By eliminating unnecessary noise, this method lays a stronger foundation for deeper text analysis or comparison.

## 4.15 ****Why Was This Update Introduced?****

* To address **entity-specific noise** in the previous results.
* To enable broader **generalization** and clearer topics by eliminating unnecessary details.
* To prepare the data for future analyses that require a **higher-level understanding** of document themes.

Code

Code

This code introduces a **comprehensive text preprocessing pipeline** which performs various text cleaning and preparation steps before further analysis. Here’s a detailed breakdown of what this code does:

## 4.16 ****Key Features of the Code:****

1. **Text Cleaning with text\_cleaner Function:**
   * **Remove @ symbols:** Any word starting with @ is removed, likely to exclude social media mentions.
   * **Substitute non-alphabetic characters:** It replaces any non-alphabetic characters (like punctuation and special characters) with spaces.
   * **Lowercasing:** All text is converted to lowercase to standardize it.
   * **Remove extra spaces:** Extra spaces are collapsed into a single space, and leading/trailing spaces are removed.
   * **Tokenization:** It breaks the text into individual words (tokens).
   * **Remove digits and non-alphabetic words:** Any digits or non-alphabetic words are removed.
   * **Remove stopwords:** The list of common English stopwords (like “the”, “and”, “is”) is excluded from the text.
   * **Lemmatization:** Words are lemmatized (reduced to their base form), considering their part-of-speech (POS).
2. **Lemmatization with POS Tagging:**
   * Each word is tagged with its **Part of Speech (POS)**, and based on the POS, the correct lemmatization process is applied.
   * Words are lemmatized into their root form (e.g., “running” becomes “run”, “better” becomes “good”).
3. **Reading .docx Files:**
   * **read\_docx function** reads the content of .docx files, extracting the text from all paragraphs and concatenating them.
4. **Iterating through Files in a Directory:**
   * The script iterates through all files in a given directory (directory variable), and for each .docx file, it reads and preprocesses the text.
   * The cleaned text is then stored in a list (texts), and filenames are stored in filenames.
5. **Displaying Cleaned Text:**
   * The cleaned text for each file is printed for verification, by this way we can visually check how the text has been processed.

## 4.17 ****Function Breakdown:****

* **text\_cleaner function** performs the core preprocessing:
  + Removes words starting with @.
  + Strips out non-alphabetical characters.
  + Tokenizes, removes digits, filters valid words, and removes stopwords.
  + Lemmatizes the remaining words and reassembles the cleaned text.
* **lemmatize function** uses POS tagging to apply the correct lemmatization based on the word’s part of speech (noun, verb, adjective, etc.).

## 4.18 ****Expected Output:****

For each .docx file in the specified directory, the program will output the cleaned text. The cleaned text will have: - No special characters, digits, or stopwords. - All words lemmatized to their base forms (e.g., “running” becomes “run”). - Words such as “better” would be lemmatized to “good.”

Here’s an example of what the output might look like:

Cleaned text from document1.docx:

this study focus on analysis of data from various sensor to predict future trends in health care.

----------------------------------------

Cleaned text from document2.docx:

ai technology is becoming increasingly important in decision making across multiple industries.

----------------------------------------

## 4.19 ****Why This Code is Useful:****

* **Data Preprocessing for NLP Models:** The cleaned text is ready for further natural language processing tasks such as topic modeling, sentiment analysis, or text classification.
* **Consistent Format:** By removing stopwords, digits, special characters, and applying lemmatization, the text becomes standardized, reducing noise and improving the accuracy of downstream models.

## 4.20 ****Potential Improvements/Modifications:****

Code

Code

Code

Code

The code I’ve provided is well-structured for cleaning and preprocessing text data, particularly from .docx files. It removes unnecessary parts of the text (e.g., session details, timestamps), performs tokenization, and filters out stop words, digits, meaningless words, and duplicates. Additionally, it performs lemmatization to ensure that the words are reduced to themy base forms.

Here’s a breakdown of how the code works:

1. **Text Cleaning (Regex Replacements):**
   * It removes specific unwanted patterns (e.g., class sessions, AM/PM timestamps) using regular expressions (re.sub).
   * Non-alphabetic characters are replaced with spaces (re.sub(r'[^0-9a-zA-Z\s]', ' ', contents)), and multiple spaces are reduced to a single space.
2. **Tokenization and Filtering:**
   * Tokenizes the text into words using nltk.word\_tokenize.
   * Filters out digits, invalid words, and stop words.
   * It also removes custom meaningless words (4bs, f, xh) and patterns like alphanumeric combinations (e.g., “ai4bs”).
3. **Lemmatization:**
   * Uses WordNetLemmatizer from NLTK to convert words to themy base form (e.g., “running” to “run”).
4. **File Handling:**
   * Reads .docx files using the docx library.
   * For each file, the text\_cleaner function is applied to preprocess the content.
5. **Dmyectory Traversal:**
   * Reads all .docx files from a specified dmyectory, processes them, and stores the cleaned content in a list.

Finally, the cleaned text from each file is printed for verification.

## 4.21 Potential Improvements:

* **Error Handling:** It might be useful to add error handling for cases where the .docx files cannot be read or processed correctly.
* **Efficiency:** If the dmyectory contains many files, I could consider processing the files in parallel or batch processing to speed up execution.

the code works as expected, I see the cleaned and processed text output from each .docx file in my dmyectory.

Code

To begin, I decided to use the textual data from the .docx files. I started by reading the files and preprocessing the text using the preprocess\_text function. The goal in this step was to clean up the text by removing noise such as punctuation or irrelevant words, converting the text into a simpler form that would be more suitable for analytical models.

After preprocessing the texts, I applied Topic Modeling to analyze the data. For this task, I used the Latent Dirichlet Allocation (LDA) model, which helped me identify different topics within the texts. To begin, I converted the texts into a document-term matrix using CountVectorizer, where each row represented a document and each column represented a word.

To ensure that the model accurately identified topics, I set the number of topics to 2 (this number can be adjusted based on the data). I selected this number because I wanted the model to extract two primary categories of concepts.

After running the model, I ended up with the following topics:

Topic 1:

[‘data’, ‘nt’, ‘income’, ‘things’, ‘people’, ‘want’, ‘time’, ‘look’, ‘lot’, ‘different’] Topic 1 is related to concepts such as “data,” “income,” and “time.” It seems to refer to the analysis of data and its use in various contexts.

Topic 2:

[‘ai’, ‘decision’, ‘based’, ‘aspects’, ‘language’, ‘database’, ‘autonomous’, ‘driving’, ‘intelligence’, ‘ultimately’] Topic 2 focuses on concepts like “artificial intelligence,” “decision-making,” and “autonomous driving.” This topic likely relates to applications of AI and related technologies.

Initially, I encountered some issues with the previous code, so I decided to implement this new approach to preprocessing and analyzing the texts. With this new method, I was able to identify the main topics present in the data and obtain meaningful results that aid in a better understanding of the texts.

Code

Here’s how I arrived at the difference in similarity scores between 89% and 47%, and the reasoning behind the process:

Initially, I started by reading and preprocessing the text data from all the .docx files. The goal was to clean the data, removing unwanted content (like session details or timestamps) and irrelevant words. This is where I focused on eliminating stop words, non-alphabetic characters, and some meaningless words like “4bs” or “f.” I also performed lemmatization to normalize the words, ensuring that they appeared in their base forms (e.g., changing “running” to “run”).

Once the preprocessing was done, I used TF-IDF (Term Frequency-Inverse Document Frequency), which is a method for transforming the text data into numerical vectors. This vectorization allows us to represent the documents in a way that makes it easier to calculate similarities between them.

For the cosine similarity step, I compared each document’s TF-IDF vector to the others. Cosine similarity measures how close two vectors are, meaning it looks at how similar the content is between the documents. The score ranges from 0 (completely different) to 1 (identical).

For instance, I initially got a similarity score of 89% between two documents (let’s call them AI2.docx and Vir1.docx). However, when revisiting the comparison later, I found a lower similarity score of 47%. This difference can be attributed to several factors:

Preprocessing: During the text cleaning process, some important words might have been removed, or the lemmatization might have altered the meaning slightly, which could have affected the final similarity scores. It’s possible that in one iteration, the preprocessing steps removed more useful words, leading to a reduced similarity score.

TF-IDF Vectorization: The TF-IDF transformation is sensitive to the presence of rare terms and their importance in a document. If two documents have more unique terms that are not shared between them, the similarity score will be lower. It’s likely that in one of the calculations, the key terms that defined the similarity between the documents were weighted differently, resulting in a significant drop.

Document Content: If the content of the documents is slightly varied, the similarity score will reflect that. For example, documents with highly overlapping words (like “ai,” “data,” “income,” “decision”) would have a higher cosine similarity, but if one document introduced more specific or different words, the similarity score could decrease.

Ranking of Keywords: The top keywords also play a crucial role. The keywords for AI2.docx include terms like “ai,” “decision,” and “probably,” which are related to decision-making and artificial intelligence. On the other hand, the keywords for Vir1.docx focus more on terms like “yeah,” “think,” and “income,” which suggest that the documents discuss slightly different topics. This difference in keyword relevance likely contributed to the observed drop in similarity between the two documents.

So, while the first result was 89%, this second calculation of 47% reflects a more accurate and nuanced similarity, taking into account the impact of preprocessing, vectorization, and the actual content of the documents.

Code

## 4.22 Key Steps Taken:

1. **Text Preprocessing**:
   * First, I read the content of each .docx file using the read\_docx function and extracted the text.
   * Then, I cleaned the extracted text by removing non-alphabetic characters and converting everything to lowercase, so that the focus was only on words without being affected by case sensitivity or non-alphabetical symbols.
   * After cleaning, I split the text into individual words using the split() function.
2. **Counting Keywords**:
   * I defined two sets of keywords: one for **AI** (keywords\_ai) and one for **vir** (keywords\_vir).
   * For each word in the cleaned text, I counted how many times words from each keyword set (AI and vir) appeared. This helped me measure the presence of AI-related and vir-related terms in each document.

## 4.23 Results:

* **File: ‘AI2.docx’**
  + AI Keywords: 82
  + Vir Keywords: 29

The AI2.docx file contains 82 occurrences of AI-related keywords and 29 occurrences of vir-related keywords. This indicates that the document focuses more on AI topics but also touches on vir-related terms.

* **File: ‘Vir1.docx’**
  + AI Keywords: 123
  + Vir Keywords: 123

In the case of the Vir1.docx file, both AI and vir-related keywords appear equally, with 123 occurrences for each. This indicates that the document equally addresses both AI and vir topics.

## 4.24 How I Reached This:

1. **Text Extraction**: I started by extracting the content of the .docx files using the python-docx library.
2. **Text Cleaning**: Then, I cleaned the text by removing unnecessary characters and splitting it into words.
3. **Keyword Comparison**: Finally, I compared the content of each document with the keyword sets for AI and vir, counting how many times each set of keywords appeared.

This method helps me quickly identify which topic—AI or vir—is more prevalent in each document and where the primary focus lies.

Code

In this code,we are calculating the similarity between documents based on the occurrence of specialized keywords related to **AI** and **vir** (visualization and storytelling) topics.

## 4.25 Explanation of the Code:

1. **Text Preprocessing**: -I read each .docx file and clean the text by removing non-alphabetical characters and converting the text to lowercase.
   * The text is then split into words for analysis.
2. **Keyword Matching**:
   * I define two sets of keywords:
     + **Keywords related to AI**: keywords\_ai
     + **Keywords related to vir**: keywords\_vir
   * For each document pair, I count the number of AI and vir keywords that appear in both documents.
3. **Similarity Calculation**:
   * For each pair of documents, I calculate the number of shared AI and vir keywords.
   * I calculate the **similarity score** as the ratio of the total number of shared keywords (AI and vir) to the total number of keywords (AI + vir) in both documents.

## 4.26 How Similarity is Calculated:

1. **Keyword Counts**:
   * For each document, I count how many times AI and vir keywords appear.
2. **Shared Keywords**:
   * The similarity between two documents is calculated based on the **shared AI** and **shared vir** keywords. The number of shared keywords is the **minimum count** of a keyword in both documents.

## 4.27 Result:

* The similarity score between **‘AI2.docx’** and **‘Vir1.docx’** is **0.2433**.
  + This means that there is a 24.33% overlap in the AI and vir-related content of the two documents based on the defined keywords.

## 4.28 How This Result is Reached:

* **Document 1 (‘AI2.docx’)**: Has 82 occurrences of AI-related keywords and 29 occurrences of vir-related keywords.
* **Document 2 (‘Vir1.docx’)**: Has 123 occurrences of AI-related keywords and 123 occurrences of vir-related keywords.
* After counting and comparing the shared AI and vir keywords, the similarity score was computed as **0.2433**, indicating a low but significant overlap in the terms related to both AI and vir between these two documents.

Code

# 5 Conclusion:

In this analysis, I have examined two distinct fields: **Artificial Intelligence (AI)** and **Data Visualization** using specialized keywords for each domain. Through the extraction and processing of keywords, I have generated word clouds that highlight the distinctions and similarities between these two fields.

* **Artificial Intelligence (AI)** focuses more on technology, algorithms, and models, with keywords like “ai”, “learning”, “machine”, “intelligence”, and “algorithm”.
* **Data Visualization** emphasizes conveying information and storytelling through data, with keywords such as “message”, “visualization”, “context”, “story”, “audience”, “insights”, and “experience”.

As a result, I have successfully highlighted the differences between these two domains through their respective keywords, providing a better understanding of both fields and their interconnections.

# 6 Suggestions for Next Steps:

1. **Expand the Keyword Sets**:
   * To enhance the accuracy of my analysis, consider expanding the keyword sets for both domains. For example:
     + In the AI domain, I could add terms like “deep learning”, “neural networks”, “automation”, and “big data”.
     + In the Visualization domain, terms like “interactive”, “graph”, “dashboard”, “chart”, and “storytelling” could be included.

Expanding these keywords will allow for more refined analyses and provide deeper insights into similarities and differences.

1. **Use More Advanced Models for Semantic Analysis**:
   * Simple keyword-based approaches are effective, but for greater accuracy, consider utilizing more advanced **Natural Language Processing (NLP)** models. Techniques such as **TF-IDF (Term Frequency-Inverse Document Frequency)** or **Word2Vec** can help me capture semantic and conceptual similarities between texts more precisely.
2. **Comparative Analysis and Clustering**:
   * For more detailed comparison and grouping of similar texts, consider using **clustering techniques** like **K-means** or **Hierarchical Clustering**. These methods can help I group similar texts together and provide a better understanding of relationships between them.
3. **Explore Relationships Between Domains**:
   * If we’re interested in deeper insights into the relationships between these two fields, I could perform **correlation analysis**. This will allow me to explore how concepts and keywords from one domain may appear in the other and how these domains might influence each other.
4. **Apply Machine Learning Models**:
   * For a more sophisticated comparison of texts, I could apply **machine learning models** like **neural networks** to analyze the texts. These models can process the data more deeply and uncover complex relationships between texts.

## 6.1 Summary:

I have laid a solid foundation for comparing the **AI** and **Data Visualization** domains based on their keywords. For the next steps, it is suggested that we expand my keyword sets, explore more advanced NLP and machine learning models, and conduct deeper semantic and comparative analyses to achieve more comprehensive and meaningful results.