DEVELOPMENT OF DISSERTATION TOPIC GENERATOR FOR M.S.C STUDENTS USING MACHINE LEARNING

**1. Loading Libraries**

* Import essential libraries for text processing, machine learning, and visualization.
* Libraries used include pandas for data handling, nltk and spacy for text preprocessing and linguistic analysis, gensim for word embeddings, sklearn for topic modeling and clustering, and visualization tools like matplotlib and seaborn.

**2. Data Collection**

* **Source**: CSV and Excel files containing primary and secondary data (e.g., student information, academic background, potential thesis topics).
* **Combination**: The data from different files are merged into a single DataFrame.
* **Cleaning**: Irrelevant columns (like Student ID, job titles, career goals) are dropped.

**3. Data Cleaning and Transformation**

* **Removing Irrelevant Columns**: Columns like 'Responsibilities' and 'Career Goals' are removed to focus on academic background and work experience.
* **Combining Columns**: Potential thesis topics are combined into one column.
* **Removing Degree Prefixes**: Use of a regex pattern to clean degree programs by removing prefixes like “B.Sc.” or “M.Sc.”
* **Handling Missing Data**: Rows with more than two missing values are dropped.

**4. Exploratory Data Analysis (EDA)**

* **Visualizing Data**: Bar plots are used to show the distribution of academic backgrounds and student interests.
* **Keyword Analysis**: Common keywords from a column (e.g., ‘Description of Social Activities’) are identified and displayed using Counter.
* **Word Cloud**: Visualization of common words in the dataset using a word cloud.

**5. Text Preprocessing**

* **Tokenization and Cleaning**: Removing punctuation, stopwords, and applying lemmatization to clean the titles.
* **Word Frequency Analysis**: Identifying the most common words in the dataset and visualizing them with bar plots.

**6. Topic Modeling (Latent Dirichlet Allocation)**

* **TF-IDF Vectorization**: Converting the cleaned text into a matrix of token counts using TfidfVectorizer.
* **LDA Modeling**: Performing Latent Dirichlet Allocation (LDA) to identify common topics in the dataset.
* **Visualization**: The topics are visualized using pyLDAvis.

**7. Sentence Structure and Syntax Analysis**

* **Dependency Parsing**: Using SpaCy for dependency parsing to analyze sentence structure.
* **POS Tagging**: Analyzing the Part-of-Speech tags for the titles to identify common syntactic patterns.

**8. Semantic Analysis using Word Embeddings**

* **Word2Vec Model**: Gensim’s Word2Vec model is applied to find semantically similar words.
* **Exploring Word Relationships**: Example words like ‘artificial intelligence,’ ‘software,’ and ‘cybersecurity’ are used to find similar terms.

This process incorporates **Bloom’s Taxonomy of Verbs** classification and a comparison of primary and secondary data through **cosine similarity** and **Word2Vec** embeddings. Here's a breakdown of the main steps and how they work:

**Step 7: Classification Using Bloom’s Taxonomy of Verbs**

* **Bloom’s Verbs Dictionary**: A dictionary is created with categories such as "Remember," "Understand," "Apply," "Analyze," "Evaluate," and "Create." Each category contains relevant action verbs.
* **Classifying Verbs**: For each thesis title, verbs are extracted and classified based on Bloom’s taxonomy. The classify\_verbs\_in\_title function identifies verbs using their part-of-speech (POS) tags and then categorizes them.
* **Summary**: A summary of the classification counts the number of verbs for each category across the dataset, giving insight into the cognitive focus of the thesis titles.

**Step 8: Cosine Similarity with TF-IDF**

* **Text Vectorization**: After cleaning and tokenizing the thesis titles, the **TF-IDF** method is used to convert the text into numerical vectors, capturing the importance of words in the context of the dataset.
* **Cosine Similarity**: The similarity between primary and secondary texts is calculated based on their TF-IDF vectors. This approach highlights which thesis topics are most similar between the two datasets.
* **Best Matches**: For each title in the secondary dataset, the most similar primary title is identified, and their similarity score is stored.

**Step 9: Using Word2Vec for Semantic Similarity**

* **Word2Vec Model**: A **Word2Vec** model is trained on the tokenized texts, capturing semantic relationships between words based on their contexts. This provides more meaningful comparisons between thesis topics than TF-IDF alone.
* **Vectorizing Text**: Both primary and secondary titles are converted into vectors by averaging the word vectors for each title.
* **Cosine Similarity with Word2Vec**: A second cosine similarity matrix is calculated using the Word2Vec embeddings, which allows for a more flexible, context-aware similarity comparison.

**Final Merged Results:**

* The result\_df2 DataFrame includes matched pairs of primary and secondary titles based on the **best semantic match** from the Word2Vec model, along with their similarity scores.
* This is complemented by the **Bloom’s Taxonomy classification**, offering insights into the cognitive complexity of the thesis titles.

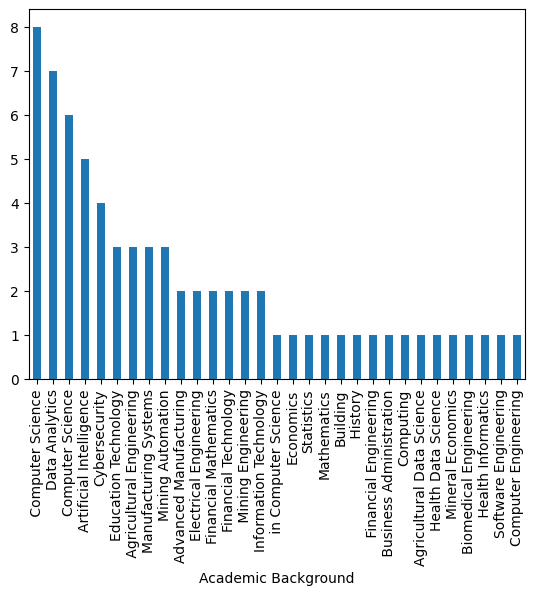
**Next Steps:**

* **Thresholding**: Fine-tune the similarity threshold to filter out weaker matches.
* **Recommendations**: Use the matched topics to generate recommendations for students based on their interests and academic background.
* **Visualization**: Create visualizations to represent the distribution of Bloom's taxonomy categories and the similarity scores.

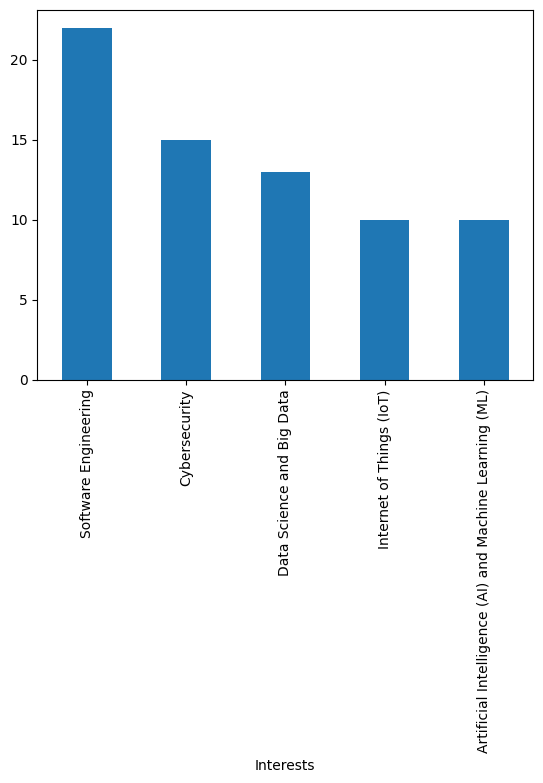
Step in methodology  
Web\_ scraping of google scholar the secondary data .

DATA EXPLORATION (PRIMARY)

COUNTPLOT FOR THE ACADEMIC BACKGROUND OF THE PRIMARY DATA



Interest (Primary Data)



Description of INTEREST



WORD CLOUD FOR THE TITTLE



Dependency Parsing Table for the various title

|  |  |  |
| --- | --- | --- |
| **Dependency Relation** | **Governor** | **Dependent** |
| nmod | machine | machine |
| acl | machine | learning |
| amod | intelligence | artificial |
| dobj | learning | intelligence |
| nummod | traveler | two |
| amod | traveler | fellow |
| nmod | machine | traveler |
| appos | traveler | quest |
| amod | machine | intelligent |
| compound | machine | behavior |
| ROOT | machine | machine |

machine: nmod -> machine: "machine" is the nominal modifier (nmod) of another "machine." This could suggest a self-modifying noun, though the context may involve a broader noun phrase.

learning: acl -> machine: "learning" is the clausal modifier (acl) of "machine." This means "learning" provides some descriptive information about "machine."

artificial: amod -> intelligence: "artificial" is the adjectival modifier (amod) of "intelligence." "artificial" describes the noun "intelligence."

intelligence: dobj -> learning: "intelligence" is the direct object (dobj) of "learning." This suggests that "learning" acts upon "intelligence."

two: nummod -> traveler: "two" is a numerical modifier (nummod) of "traveler." This means "traveler" is being quantified by "two."

fellow: amod -> traveler: "fellow" is an adjectival modifier (amod) of "traveler." It provides descriptive information about "traveler."

traveler: nmod -> machine: "traveler" is a nominal modifier (nmod) of "machine," implying that "traveler" is related to or modifying "machine."

quest: appos -> traveler: "quest" is an appositive (appos) of "traveler." This suggests that "quest" renames or further explains "traveler."

intelligent: amod -> machine: "intelligent" is an adjectival modifier (amod) of "machine." "intelligent" describes "machine."

behavior: compound -> machine: "behavior" is a compound noun, forming part of the noun phrase with "machine."

machine: ROOT -> machine: "machine" is the root of the sentence. It serves as the central verb or noun from which other elements depend.

Analyzing POS Patterns

**Understanding the POS Tags:**

* **JJ:** Adjective
* **NN:** Noun
* **VBG:** Verb, Gerund or Present Participle
* **VBN:** Verb, Past Participle
* **RB:** Adverb
* **NP:** Noun Phrase

**Analyzing the POS Patterns:**

Based on the provided POS patterns and the sentence, we can identify common syntactic structures. Here's a breakdown in tabular form:

|  |  |  |
| --- | --- | --- |
| **Pattern** | **Example from Sentence** | **Structure** |
| JJ NN NN NN | intelligent machine learning behavior | Adjective Noun Noun Noun |
| JJ NN NN VBG JJ NN NN JJ NN | two fellow traveler quest intelligent machine learning behavior | Adjective Noun Noun Verb, Gerund Adjective Noun Noun Adjective Noun |
| JJ NN NN JJ NN VBN RB NN VBG NN NN | intelligent machine learning artificial intelligence developed in Nigeria | Adjective Noun Noun Adjective Noun Verb, Past Participle Adverb Noun Verb, Gerund Noun Noun |
| JJ NN NN | intelligent machine learning | Adjective Noun Noun |
| JJ NN NN VBG JJ NN | two fellow traveler quest intelligent machine | Adjective Noun Noun Verb, Gerund Adjective Noun |
| JJ NN NN VBG JJ JJ NN | intelligent machine learning artificial intelligence | Adjective Noun Noun Verb, Gerund Adjective Adjective Noun |
| NN VBG JJ NN NN NN | machine learning artificial intelligence developed in Nigeria | Noun Verb, Gerund Adjective Noun Noun Noun |
| NN NN NN NN | machine learning artificial intelligence | Noun Noun Noun Noun |
| VBG NN NN NN | learning artificial intelligence | Verb, Gerund Noun Noun Noun |
| VBN NN NN | developed in Nigeria | Verb, Past Participle Noun Noun |

Summary of Verb Classification According to Bloom's Taxonomy

|  |  |
| --- | --- |
| Bloom's Taxonomy Level | Verb Count |
| Unknown | 530 |
| Understand | 1 |
| Remember | 1 |
| Apply | 0 |
| Analyze | 0 |
| Evaluate | 0 |
| Create | 0 |

Based on Human Evaluation on the three Vectorization , SentenceTransformer is the best

Using ANN MODEL DEVELOPMENT

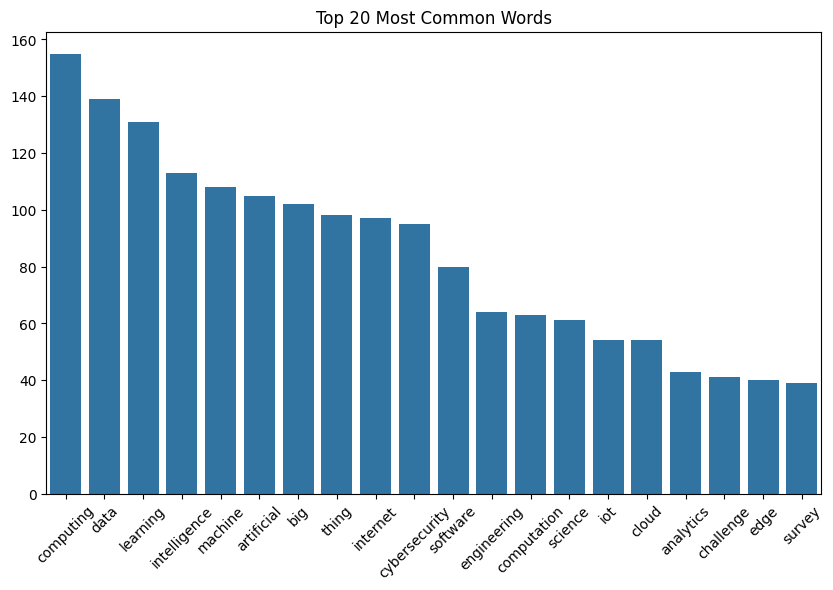
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Epochs | CPU Time (seconds) | Total Parameters | MSE | RMSE | MAE | R-Squared |
| TfidfVectorizer | 10 | 7.41 | 25,920 | 0.00571 | 0.07559 | 0.01951 | 0.04366 |
| TfidfVectorizer | 50 | 25.64 | 25,920 | 0.0059 | 0.07679 | 0.01795 | 0.04037 |
| TfidfVectorizer (Model 3) | 50 | 33.92 | 91,232 | 0.00706 | 0.08404 | 0.03009 | 0.01054 |
| TfidfVectorizer (Hypetuned Model) | 50 | 6.17 | 65,024 | 0.00572 | 0.07561 | 0.01962 | 0.04367 |
| Word2Vec | 10 | 9.16 | 25,920 | 0.00157 | 0.03959 | 0.02614 | -0.06183 |
| Word2Vec | 50 | 16.2 | 25,920 | 0.00156 | 0.03955 | 0.02858 | -2.15288 |
| Word2Vec (Hypetuned Model) | 50 | 5.88 | 15,956 | 0.00148 | 0.03853 | 0.02532 | -0.06498 |
| Sentence Transformer | 10 | 9.38 | N/A | 0.12792 | 0.35765 | 0.28139 | 0.05822 |
| Sentence Transformer | 50 | 32.3 | N/A | 0.12792 | 0.35765 | 0.28139 | 0.05822 |

TfidfVectorizer (Hypetuned Model) with 50 epochs is the most balanced choice for a recommendation system. It offers good performance with reasonable error rates and fast CPU time.

Word2Vec, while efficient in terms of CPU time, has an undesirable R-Squared, indicating that it might not generalize well.

SentenceTransformer is not suitable due to high error metrics and longer CPU time.

Therefore, TfidfVectorizer (Best Model) would be the best option for your recommendation system.



Data Cleaning and Exploration