An Application Oriented Text & Web Mining Project

ON

Focused Web Crawler

DONE BY

Nitin Kankanala (002705445)
Srinivas Narne (002705961)

Abstract:

Document similarity analysis is a fundamental task in natural language processing and information retrieval. This project aims to address the challenge of comparing a seed document with a collection of other documents extracted from a specific source, such as Wikipedia. The project begins by taking user input, typically a keyword or search term, which serves as the seed document. The corresponding page or section from the chosen source is retrieved, and a set of documents linked within that page is extracted. The objective is to find meaningful representations for both the seed document and the extracted documents. To achieve this, various techniques for generating document embeddings are explored. These embeddings convert the textual data into numerical vectors, allowing for direct comparisons between documents based on their similarity scores. Once the embeddings are obtained, the project calculates similarity scores between the seed document and each of the extracted documents. These scores provide valuable insights into the relatedness of the documents, helping to identify the most relevant matches. Moreover, the project may explore advanced methods like clustering and visualization techniques to gain a deeper understanding of the relationships among documents and potentially discover patterns or trends. The success of the project will be evaluated based on the accuracy and relevance of the document similarity analysis, as well as the efficiency of the chosen techniques in handling large datasets. In conclusion, this project contributes to the field of document similarity analysis and information retrieval, offering valuable tools for organizing and accessing information from diverse sources in a meaningful and efficient manner.

Keywords- ***web crawler, BERT, seed page, website links, Embeddings, Cosine Similarity, natural language processing, semantic representations***

Introduction:

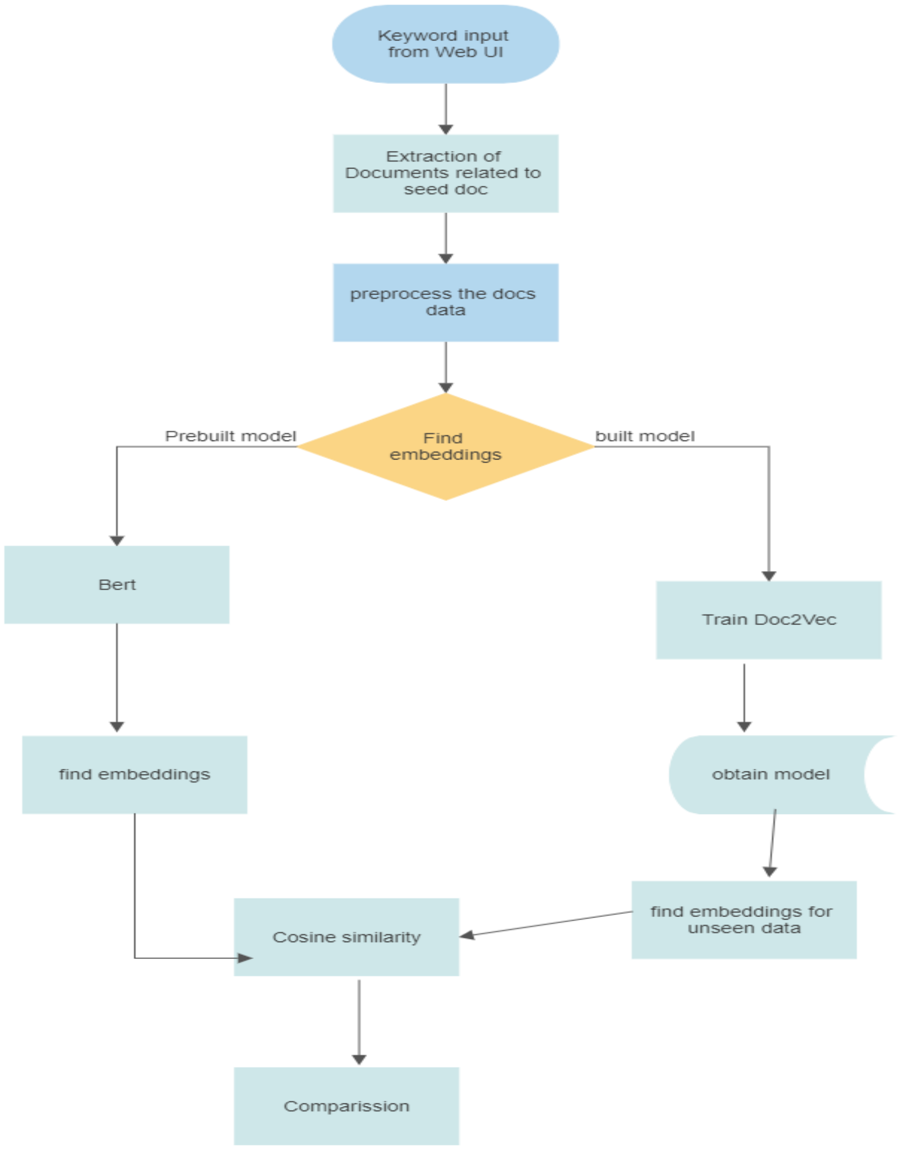
In the realm of natural language processing (NLP) and information retrieval, document similarity analysis is a vital undertaking. With the exponential growth of digital content from various sources, such as websites, articles, and databases, the need to assess and compare the relatedness of documents has become increasingly crucial. Document similarity analysis plays a pivotal role in content clustering, information retrieval, plagiarism detection, and knowledge discovery, providing valuable insights and efficient navigation through vast amounts of textual data. The primary aim of this project is to design and implement a robust system for document similarity analysis, specifically targeting the comparison between a seed document and a collection of other documents extracted from a chosen source, such as Wikipedia. The seed document is typically obtained through user input, often in the form of a search term or keyword, defining the topic of interest. The first step of the project involves retrieving the relevant page or section from the chosen source that corresponds to the seed document. From this page, a set of documents linked within the content is extracted, forming the basis for comparison. To achieve meaningful and efficient document comparisons, various techniques for generating document embeddings are explored. Document embeddings are numerical representations of textual data, allowing for quantitative assessments of document similarity. Two prominent embedding techniques under investigation are BERT embeddings and Doc2Vec embeddings. BERT embeddings utilize transformer-based architectures to capture contextual information, while Doc2Vec embeddings leverage paragraph vector representations to capture semantic meaning. Once the embeddings are obtained, the system computes similarity scores between the seed document and each of the extracted documents. These similarity scores provide quantitative measures of the relatedness between documents, helping to identify the most relevant matches. To enhance the analysis, the project may delve into advanced techniques such as clustering algorithms and visualization methods. Clustering algorithms group similar documents together, providing structured representations of the corpus. Visualization techniques offer intuitive ways to portray document relationships, aiding users in identifying patterns and connections. The success of the project hinges on the accuracy, efficiency, and scalability of the document similarity analysis. Additionally, it seeks to shed light on the effectiveness of various embedding techniques in real-world scenarios and their potential impact on information retrieval and content organization. In conclusion, this project aspires to contribute to the domain of document similarity analysis and NLP by providing researchers, data scientists, and information seekers with powerful tools for managing and analysing vast amounts of textual data. By enabling accurate and meaningful document comparisons, this system holds the promise to revolutionize the way users interact with textual information, empowering them with valuable insights and facilitating more informed decision-making processes.

Literature Review:

The literature review showcases the extensive research efforts in document similarity analysis and text embeddings. Document similarity analysis plays a pivotal role in diverse applications, including content clustering, information retrieval, and knowledge discovery. Different embedding techniques have been proposed to generate numerical representations of documents, enabling efficient comparisons and similarity measurements. The Doc2Vec model, introduced as an extension of Word2Vec, provides dense vector representations for entire documents, capturing semantic meaning and context. BERT embeddings, based on transformer architectures, offer context-aware language models that significantly improve the understanding of sentence-level semantics. To address efficiency challenges, techniques like Locality-Sensitive Hashing (LSH) have been employed, enabling quick identification of candidate pairs of similar documents in large-scale collections. Graph-based representation learning methods, such as Deep Walk, leverage document relationships to generate meaningful embeddings. Additionally, the Universal Sentence Encoder presents a cross-lingual and cross-domain approach to sentence embeddings, further enriching the field of document similarity analysis. By building upon these foundational works, our project aims to develop a comprehensive system for document similarity analysis. We explore the application of BERT and Doc2Vec embeddings, along with various similarity measures, to enhance the accuracy and efficiency of document comparisons. Through this research, we endeavour to contribute to the advancement of information retrieval and content organization, empowering users with valuable insights and efficient access to vast amounts of textual data.

Methodology:

This project aims to address the challenge of comparing a seed document with a collection of other documents extracted from a chosen source, such as Wikipedia. The seed document is obtained through user input, typically in the form of a search term or keyword. By leveraging two prominent embedding techniques, BERT and Doc2Vec, this project seeks to generate meaningful vector representations for the documents and compute similarity scores to quantify their relatedness. The key procedure followed is extraction of documents related to a seed document and finding the embeddings that are used for finding the similarity scores of each extracted document and main seed document. Then we have built our own model for finding out the embeddings by training on collected data on the go processed in the backend and then take unseen data to find the embeddings and find the similarity score for those new data. Also a comparison of the similarity score found for the seed document using Bert model's embeddings and this doc2vec approach are shown to see how well our model is performing. Let’s have a detailed look of what is happening in the procedure…



Flow Chart of the process done

Data Collection:
In the data collection phase, the project gathers input from users through a text box, which serves as the search word or keyword for the seed document. A web scraping technique is employed to retrieve the relevant Wikipedia page corresponding to the seed document. From this page, a collection of 50 linked documents is extracted to form the dataset for comparison. Selenium Library is used to extract the data from the Wikipedia pages. After taking the keyword input from the user the process is automated to extract the data from the Wikipedia source where initially the seed document is extracted and we get the links in that seed page for which in later steps embeddings are found to get the similarity scores of how relevant they are to the source seed document.

Data Preprocessing:
Prior to performing similarity analysis, the data is pre-processed to ensure consistency and enhance the quality of embeddings. Text preprocessing tasks, such as tokenization, removing stop words, and converting text to lowercase, are applied to both the seed document and the extracted documents. The Wikipedia data is pre-processed as a general process and given as input for the models.

BERT Embeddings:
BERT, a powerful language model, is utilized to generate vector representations (embeddings) for the seed document and the 50 extracted documents. Pre-trained BERT models are used for this purpose, capturing contextual information and semantic meaning within the documents. By computing similarity scores between the BERT embeddings of the seed document and each of the extracted documents, the relatedness of the documents is quantified.

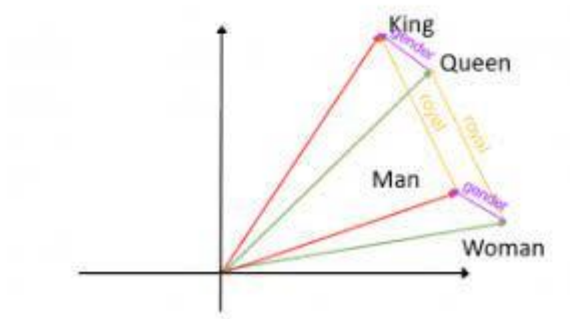
Understanding BERT in-detail
BERT, which stands for Bidirectional Encoder Representations from Transformers, is a state-of-the-art natural language processing (NLP) model introduced by Google in 2018. Unlike traditional language models that process text in a unidirectional manner, BERT uses a bidirectional transformer architecture.

The key innovation of BERT lies in its pre-training process. It is first trained on a large corpus of text using two unsupervised tasks: masked language modelling (MLM) and next sentence prediction (NSP). In MLM, random words are masked in a sentence, and the model is tasked with predicting the masked words. This fosters a deep understanding of word relationships and context. In NSP, the model is trained to predict whether two sentences in a pair are contiguous in the original text. This helps BERT to capture document-level semantics and relationships between sentences.

BERT's pre-training allows it to learn powerful contextual embeddings for words, making it suitable for a wide range of downstream NLP tasks, such as text classification, question answering, sentiment analysis, and document similarity analysis. Fine-tuning is the process of adapting BERT to specific tasks by training it on labeled data for those tasks.

Overall, BERT has become a foundation for many state-of-the-art NLP models and continues to advance research and applications in language understanding and generation. Its ability to capture deep contextual relationships has significantly improved the performance of various NLP tasks and opened up new possibilities for language understanding.

Similarity Score Calculation:
For the embeddings from the BERT model the data is fed to a function called cosine similarity which gives how similar are two quantities.
Cosine similarity is a metric used to measure the similarity between two vectors in a multi-dimensional space. In the context of natural language processing and document similarity analysis, cosine similarity is often employed to compare the similarity between document embeddings or word embeddings.
Given two vectors A and B, the cosine similarity is calculated as the cosine of the angle between the two vectors, which can be represented as:



Cosine similarity representation

In a simplified 2-dimensional space, let's consider the word embeddings for "king," "queen," "man," and "woman." The word embeddings are represented as vectors with two values each, reflecting their positions in this space. The vector for "king" is [1, 0], "queen" is [0, 1], "man" is [1, 1], and "woman" is [0, 2].

To calculate the cosine similarity between word pairs, we use the formula: cosine_similarity = (A · B) / (||A|| * ||B||), where A and B are the vectors being compared, and ||A|| and ||B|| are their respective Euclidean norms.

The cosine similarity between "king" and "queen" is calculated to be 0, indicating that they are orthogonal and have no similarity in this 2-dimensional space. On the other hand, the cosine similarity between "man" and "woman" is found to be 0.5, implying some level of similarity between these two words in this simplified space where (A · B) denotes the dot product of vectors A and B, and ||A|| and ||B|| represent the Euclidean norms of vectors A and B, respectively.

In document similarity analysis, cosine similarity is used to compare document embeddings (e.g., BERT or Doc2Vec embeddings) and assess how similar two documents are in terms of their semantic content. Documents with higher cosine similarity scores are considered more similar, while those with lower scores are less related. It is more robust to the curse of dimensionality, which refers to the increased sparsity and computational complexity of data as the number of dimensions grows. Cosine similarity measures the relative orientation of vectors, making it insensitive to the scale of the vectors and well-suited for capturing semantic relationships between words and documents.

Doc2Vec Model Training:
Doc2Vec, also known as Paragraph Vector, is an extension of the popular Word2Vec model that learns fixed-size vector representations (embeddings) for entire documents or paragraphs. While Word2Vec focuses on generating embeddings for individual words, Doc2Vec takes into account the context of entire documents, capturing their semantic meaning and preserving their coherence. The training process for the Doc2Vec model involves associating each document with a corresponding tag or label. These tags act as unique identifiers for the documents during training. There are two main approaches to training the Doc2Vec model:

Distributed Memory incorporates both document embeddings and word embeddings. During training, it predicts the next word in a sentence based on the context of both the words within the sentence and the document's embedding. The document's embedding is treated as an additional word in the context window, allowing the model to capture document-level semantics.

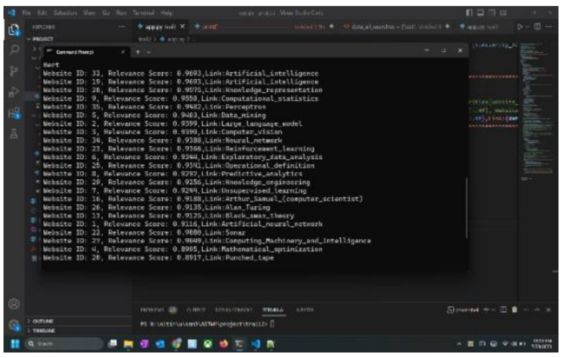
Distributed Bag of Words Model focuses on learning document embeddings. During training, it predicts a random set of words within the document based solely on the document's embedding, without considering the context of individual words. This approach is computationally less intensive compared to the DM model and may be more suitable for large datasets.

In both cases, the training process uses stochastic gradient descent or other optimization techniques to adjust the embeddings' values, minimizing the difference between predicted and actual word (and document) representations. By training the model on a large corpus of diverse documents, Doc2Vec learns to generate meaningful embeddings that capture the semantic information and relationships between different documents.

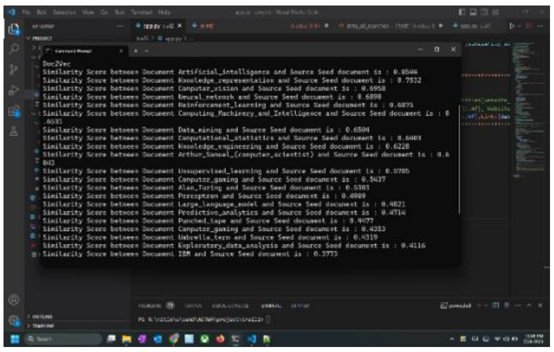
Once the Doc2Vec model is trained, it can be used to generate embeddings for new, unseen documents. This is achieved through the infer-vector method, which allows the model to infer document embeddings based on the trained parameters and context learned during training. In summary, Doc2Vec is a powerful model for learning embeddings that represent entire documents. Through training and inference, it enables us to capture the semantic meaning of documents, facilitating various natural language processing tasks, including document similarity analysis, content recommendation, and clustering.

Generating Doc2Vec Embeddings:
Generating Doc2Vec embeddings is the process of converting new, unseen documents into numerical representations that capture their meaning and context. These embeddings are fixed-size vectors that encode the semantic information of the documents. To do this, we use a trained Doc2Vec model, which has already learned how to understand the relationships between words and documents from a large dataset. The model takes the new documents as input and outputs the corresponding embeddings. These embeddings can be used for tasks like finding similar documents, organizing content, and making recommendations without having to retrain the entire model. It allows us to work efficiently with new text data while leveraging the knowledge acquired during the initial training.

Doc2Vec Embeddings Comparison:
Similar to the BERT embeddings comparison, similarity scores are computed between the Doc2Vec embeddings of the seed document and each of the extracted documents. These scores offer additional insights into the relatedness of the documents, allowing a comparison with the similarity scores obtained from BERT embeddings.
The output is as follows;



```
Bert
Website ID: 32, Relevance Score: 0.9693,Link:Artificial_intelligence
Website ID: 19, Relevance Score: 0.9693,Link:Artificial_Intelligence
Website ID: 28, Relevance Score: 0.9575,Link:Knowledge_representation
Website ID: 9, Relevance Score: 0.9550,Link:Computational_statistics
Website ID: 35, Relevance Score: 0.9482,Link:Perceptron
Website ID: 5, Relevance Score: 0.9403,Link:Data_mining
Website ID: 2, Relevance Score: 0.9399,Link:Large_language_model
Website ID: 3, Relevance Score: 0.9398,Link:Computer_vision
Website ID: 34, Relevance Score: 0.9388,Link:Neural_network
Website ID: 23, Relevance Score: 0.9366,Link:Reinforcement_learning
Website ID: 6, Relevance Score: 0.9344,Link:Exploratory_data_analysis
Website ID: 25, Relevance Score: 0.9341,Link:Operational_definition
Website ID: 8, Relevance Score: 0.9292,Link:Predictive_analytics
Website ID: 29, Relevance Score: 0.9256,Link:Knowledge_engineering
Website ID: 7, Relevance Score: 0.9244,Link:Unsupervised_learning
Website ID: 16, Relevance Score: 0.9188,Link:Arthur_Samuel_(computer_scientist)
Website ID: 26, Relevance Score: 0.9135,Link:Alan_Turing
Website ID: 13, Relevance Score: 0.9125,Link:Black_swan_theory
Website ID: 1, Relevance Score: 0.9116,Link:Artificial_neural_network
Website ID: 22, Relevance Score: 0.9080,Link:Sonar
Website ID: 27, Relevance Score: 0.9049,Link:Computing_Machinery_and_Intelligence
Website ID: 4, Relevance Score: 0.8995,Link:Mathematical_optimization
Website ID: 20, Relevance Score: 0.8917,Link:Punched_tape
```



```
Doc2Vec
Similarity Score between Document Artificial_intelligence and Source Seed document is : 0.8546
Similarity Score between Document Knowledge_representation and Source Seed document is : 0.7532
Similarity Score between Document Computer_vision and Source Seed document is : 0.6958
Similarity Score between Document Neural_network and Source Seed document is : 0.6898
Similarity Score between Document Reinforcement_learning and Source Seed document is : 0.6875
Similarity Score between Document Computing_Machinery_and_Intelligence and Source Seed document is : 0
.6531
Similarity Score between Document Data_mining and Source Seed document is : 0.6504
Similarity Score between Document Computational_statistics and Source Seed document is : 0.6403
Similarity Score between Document Knowledge_engineering and Source Seed document is : 0.6228
Similarity Score between Document Arthur_Samuel_(computer_scientist) and Source Seed document is : 0.6
043
Similarity Score between Document Unsupervised_learning and Source Seed document is : 0.5785
Similarity Score between Document Computer_gaming and Source Seed document is : 0.5437
Similarity Score between Document Alan_Turing and Source Seed document is : 0.5303
Similarity Score between Document Perceptron and Source Seed document is : 0.4989
Similarity Score between Document Large_language_model and Source Seed document is : 0.4821
Similarity Score between Document Predictive_analytics and Source Seed document is : 0.4714
Similarity Score between Document Punched_tape and Source Seed document is : 0.4477
Similarity Score between Document Computer_gaming and Source Seed document is : 0.4353
Similarity Score between Document Umbrella_term and Source Seed document is : 0.4319
Similarity Score between Document Exploratory_data_analysis and Source Seed document is : 0.4116
Similarity Score between Document IBM and Source Seed document is : 0.3773
```

Bert Model Scores & Doc2Vec Scores for the search keyword machine learning

Doc2Vec Scores

Key Description:

The Working flow of the application is described here. First the main page is used to take input from user to search a specific topic from Wikipedia and get the content of related and documents along with similarity scores.

Data Crawling:

The crawling of the data is based in the keyword the user enters in the given text box and the data is extracted from the Wikipedia pages using selenium web driver in Firefox browser. There is a limit on how deep we can extract the seed document and then all those links data will be extracted and then obtain the documents paragraphs in a data structure.

Bert Model:

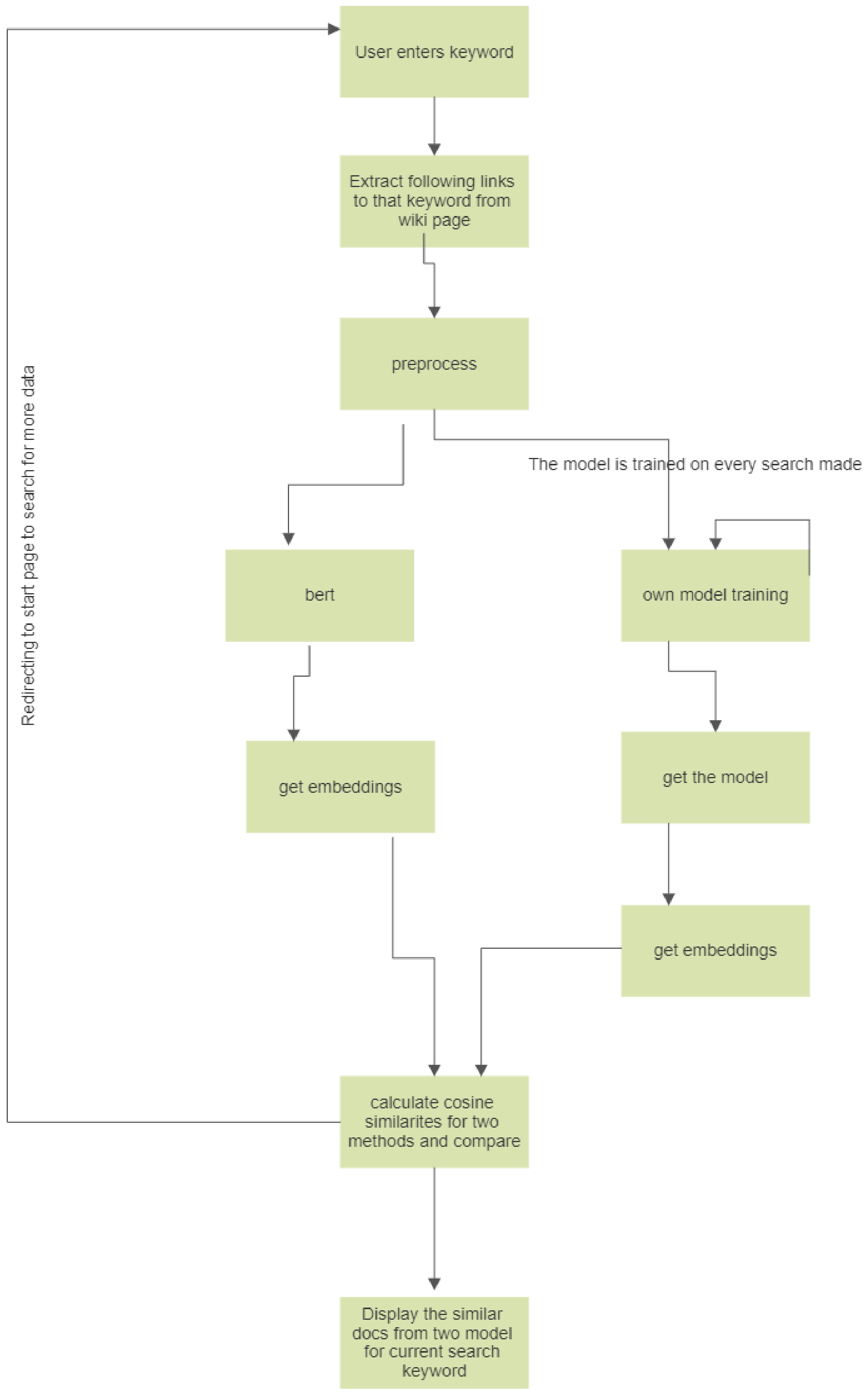
The BERT model is used to get embeddings and then similarity scores is calculated by using cosine similarity and the current search keywords related documents is displayed in the web page.

Building Model:

We used doc2Vec model to train on the data of what we extracted so far and then get a model which is in the normal flow of the code. Every search except for the first search the model is used to get embeddings and then calculate the cosine similarity at each search then display it for the current search. At every search the model is trained iteratively and as you search for more keywords the better the model will be.

Note this is not an application to get input all keywords (for like 100 searches) at once and train the model but train the model after every search. We have developed this step because we just wanted to Identify how the model is performing at each search. The data of similarity scores is shown in the console.

In the code We have mentioned clear comments for each task as the place in this report is exceeding 10 pages we just skipped the code snippets in this report.


User enters keyword
Extract following links to that keyword from wiki page
preprocess
The model is trained on every search made
bert
own model training
get embeddings
get the model
get embeddings
calculate cosine similarites for two methods and compare
Display the similar docs from two model for current search keyword
Redirecting to start page to search for more data

Flow of data

Results:

Interpreting the different similarity scores obtained from BERT and Doc2Vec models requires considering various factors and understanding the characteristics of both approaches. Here are our observations on how the two models are evaluated.

Contextual Understanding of Bert is not good:
BERT is a contextualized language model that captures rich context and semantics. When documents are highly similar or share common phrases with the seed document, BERT's ability to understand context might lead to higher similarity scores. However, high scores may also indicate potential overfitting if the model memorizes exact matches from the training set. In our application the data has similarity scores of around 90% for all the docs for a search
Which can be shown as overfitting.

Implemented Model Interpretability is better for Content:
Doc2Vec provides more transparent and interpretable embeddings compared to BERT. Each dimension in the Doc2Vec embeddings has a clear meaning, making it easier to understand why certain documents are similar or dissimilar. This interpretability can be advantageous when you need to explain the relevance assessment to stakeholders or make informed decisions based on similarity patterns. The similarity scores are verified manually by checking the high score docs and their content.

To gain deeper insights, you could perform further analysis, such as:
Fine-tuning the Doc2Vec model on a larger and more diverse dataset to potentially improve the quality of embeddings and relevance estimation. Implementing other similarity metrics, such as Euclidean distance or Manhattan distance, to compare their performance with cosine similarity. Visualizing the embeddings in a lower-dimensional space using dimensionality reduction techniques like t-SNE to explore the relationship between documents visually.

Contributions:

Our team contributions are as follows:
Nitin Kankanala: The Crawling using selenium has been implemented by him. Along with that the bert model usage to find embeddings and Doc2Vec Model Building is done by him.
Priyanka Ghare: The Data Extraction , preprocessing along with the Similarity calculation using the cosine similarity is implemented by her. The data pipeline and flask application is built by her.