INTELLIGENT NLP SEARCH BOT ON CLOUD

Author: Gayatri Rajmane, Johndeere

ABSTRACT

Nowadays, people rely on search engines and chatbots to retrieve and share information from various resources. Search can be carried out mainly in two ways, keyword search and semantic search. In a simple search, searching is carried out word to word, while in the keyword search, the upper case, the lower case, and order is ignored, and the specific results having the exact words are returned. Semantic search is a searching technique where the meaning of the searched query is being looked for, along with its context, so that the most relevant searches can be obtained with maximum information. Unlike the keyword search, which provides the exact word results without knowing its context and displaying limited insights to the users. The search engines do not currently offer accurate search with the best relevance as it is based on keyword search and not semantic search.

To put this into perspective if the user searches, "what is my destination pickup point? "Keyword search would try to return the sentences having the exact same words from the dataset, ignoring the sequence sensitivity. While if the same input is fed to the search engine using the semantic search technique, it will return all the results with a similar meaning like "last stop", "address to drop", etc., and the keyword search. This gives the users a more accurate and broader scope of insights for the searched query.

These ambiguities from the results are reduced using the semantic search approach by filtering out the results that have less relevance with respect to the context of the query. The broad range of my project includes Artificial intelligence, Semantic search, and Natural Language Processing techniques to give the highest relevant search result based on domain knowledge (keywords). The focus of the current project is on the following areas:

- **Natural language processing**: To extract the correct information.
- **Semantic Search**: To find the relation between search and the data.
- **Integration**: To develop a chatbot for interactive sessions

Key Words: Artificial Intelligence; Semantic Search; Natural Language Processing (NLP); Amazon Web Services (Aws) Cloud; NLP Chatbot.
Project Areas: Machine Learning, Cloud.

1. INTRODUCTION

High availability of services, zero wait time, reduced manual intervention, time and cost of the resources, interactive searching is achieved by implementing the smart search on AWS Cloud. This smart search agent helps retrieve results that are more relevant to the user's interest by reducing the cognitive load and increasing the effectiveness of the search. An intelligent agent would act as an intermediary between the user and the web search engine.

The semantic search would have no cognitive load and save the time of the users. The proposed work on Semantic search is accomplished by Latent Semantic Indexing (LSI) using Natural Language Processing (NLP). Using LSI Model, the query searched by the user is processed by the algorithm, i.e., the meaning and the context of the searched item are understood, and the highest relevance solution is provided to the users. Before the LSI Model gets the query data, feature extraction is performed on the data using the TD-IDF (Term Frequency – Inverse Term Frequency) Model to reflect how important the words are for further analysis. As machines do not understand the raw data provided as humans do, it needs to be converted into a numerical format that is easily understood by the machine. And to convert this data to machine-understandable format, TD-IDF Model is being used to convert text into numeric vectors. The Term Frequency model measures how frequently a term appears in the document or the query from the user. While the IDF (Inverse Document Frequency) measures the importance of the term because just computing the TF won't completely understand the importance of the query. The output of the TF-IDF Model is passed as the input to the LSI Model to understand the context of the search and retrieve the best relevant results for the user's search.

The Latent Semantic Indexing (LSI) model of Natural Language processing is one of the topic modelling techniques which creates structured data from unstructured data collection. LSI is used here to understand the search context as the same words can have different meanings in different sentences. In this way, the best relevant search results can be obtained for the users.

1.1 LITERATURE REVIEW

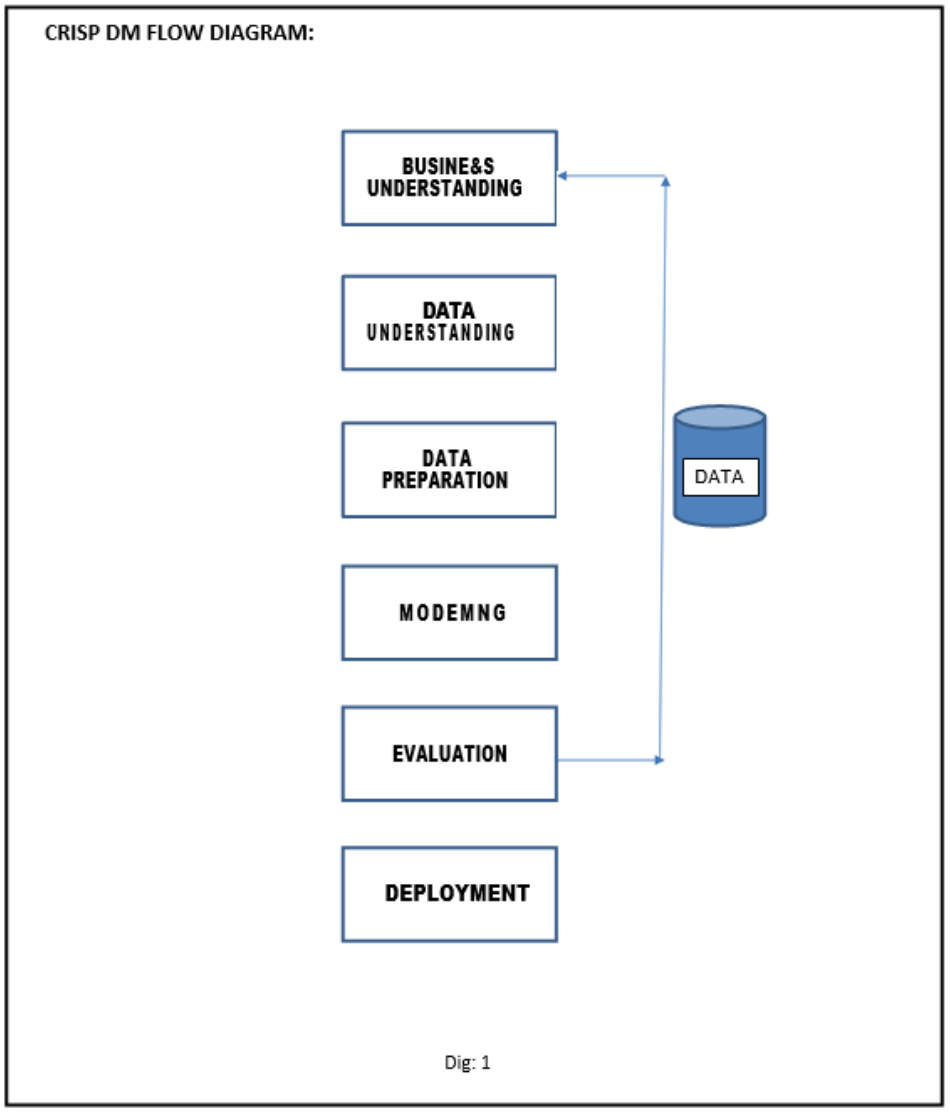
In today's period of technological advancement and innovation, people are becoming increasingly dependent on search engines to seek information and solutions, which has become an essential part of their lives. Industries require searching tools to gain insights from unstructured data for business intelligence and improve operational efficiency. For small businesses, the search is an essential factor in speeding up their business by obtaining the proper ranking through search engines.

Searching in different scenarios saves time by avoiding going through the data available previously and finding the most relevant solution for the searched query. At the same time, searching for information on the web could also be stressful for the user and increase the cognitive load. To avoid the load and manual interventions, searching can be done in two ways, keyword search and semantic search. Keyword search is limited to only those exact words in the searched query by the user and does not include the relevant contextual insights. Hence, relying on keyword search alone will not be able to achieve the best possible results. Unlike semantic search, which leverages Artificial Intelligence, saves time and cost and acts as a bridge between humans and machines by understanding the meaning of the searched query by the user and provides the most relevant results filtering out the ambiguities with

maximum informational insights.

The CRISP-DM (Cross Industry Standard Process for Data Mining) process for developing the data science life cycle consists of six phases. The Life cycle has guidelines for planning, organizing, and executing data science projects. CRISP DM has the following six stages, which are implemented while carrying out the project:

- **Data Understanding**: Collecting the initial data, exploring the data and verifying the quality of data present and operations to be performed.
- **Data Preparation**: Preprocessing the data that is cleaning, selecting, and constructing the data, including integrating and formatting data based on the business understanding.
- **Modelling:** Building suitable models and generating test outputs.
- **Evaluation:** Evaluate the process, review, and describe the next steps.
- **Deployment:** Reviewing the project, testing the expected outcome after running the model with user-friendly access. This methodology helps determine the task without clutter and with a proper execution flow.



1.3 IMPLEMENTATION:

1.3.1 Data Understanding & Preparation: Data Strategy

- **Encoding Technique Implementation**
- **Text Data Cleaning, Pre-processing & Tokenizing**

The initial cleaning and preprocessing steps play a vital role in analysis and are considered important. Cleaning the data consists of and is required here are:

- Removing distracting single quotes
- Removing Words containing digits
- Deleting Extra spaces
- Removing Unwanted lines
- Removing Punctuations
- Lemmatizing

  Then, transforming the raw data into an understandable format is called preprocessing. This process, if carried out properly, will reduce the ambiguous results.

  - Stop Words
  - Lemmatization is performed on the word text to avoid unnecessary endings and bring the word into base form. Example – Changing -> Change (after lemmatization – original form)
  - Tokenizing is converting the text into a tokens process. And tokens are derived in the following process:
  - Corpus -> Documents -> Paragraphs -> Sentences -> Tokens
  - Filtering

Fig: Data Cleaning, Preprocessing & Tokenizing

For applying the preprocessing and tokenizing process on each record of the required columns, **the. map** function of Python is used like a for loop iteration till the end of each columns record.

This ensures that no data remains ambiguous and not tokenized.

1.3.2 Modeling: Data Analysis & Evaluation Modelling

- **WordCloud Visualization**
- **Build Dictionary:**

Gensim (Generate Similar) is a python library used for Natural Language Processing (NLP). Dictionary is used to store the data in a key-value format for faster retrieval. At the same time, the corpus contains the text related to the dataset provided in a document format.

```
In [25]: from gensim.corpora import Dictionary
         dictionary = Dictionary(df_final["allTokens"])
```

Fig: Build Dictionary

- **Word Frequency:**

  The word frequency uses the above-tokenized list of the data to calculate the frequency of each word in every sentence. Here the value is the word frequency dictionary, and the key is the sentence itself. This output of word frequency acts as a parameter to the TD-IDF model.

- **TD-IDF Model (Term Frequency- Inverse Document Frequency):**

Making machines understand the text is a challenging task that persists. As computers do not understand the text in raw format. There are two techniques to convert the text into a numeric vector (numerical form), Bag of Words and TF-IDT (Term frequency – Inverse Term Frequency)

A few prerequisites are required before vectorizing the sentence. The resulting vector should not be the sparse matrix as the computation cost is higher and should contain the most linguistic information present in the sentence.

TF (Term Frequency) model measures how frequently a term occurs in a particular corpus by using a formula as follows:

$$tf_{t,d} = \frac{n_{td}}{Number\ of\ terms\ in\ the\ document}$$

The numerator in the above formula is the total number of terms 't' that appears in a corpus or document.

Here t = term and d = document.

For example, consider the sentence "AWS is a good cloud provider."

Here, vocabulary is – AWS, is, cloud, good, a, provider, Azure, resources (considering other sentences in the document)

Number of words in the sentence: 6

TF for the word AWS: (Number of times AWS occurs in a sentence)/ (number of terms in the sentence) = 1/6

Similarly, TF is calculated for the terms in the corpus.

As Term Frequency (TF) only gives the frequency of the word, not its importance in the context of the To obtain the context of the document **IDF (Inverse Document Frequency)** model is useful. IDF plays a vital role because it measures how important a term is, while TF alone cannot determine the importance of the words. IDF uses the following formula to determine the significance of the words:

$$idf_t = \log \frac{Number\ of\ documents}{Number\ of\ documents\ with\ the\ term\ 't'}$$

For Example: Consider the sentence "AWS is a good cloud provider."

IDF('AWS') = log (number of documents/number of documents containing the word 'AWS') = log (3/1) = log (3) = 0.48 (Considering there are 3 cloud provider sentences and AWS has occurred only once in the sentence provided)

Similarly, IDF values are calculated for whole built vocabulary. Hence words like "this", "is", "and" etc. are ignored as the log will be the lowest of these words and have little importance. And more important words have higher values.

After the TF and IDF values are computed individually, the TF-IDF score needs to be computed. Scores with higher values have more importance, and fewer values indicate less importance. The below formula calculates the combined score for TF and IDF values:

$$(tf_{idf})_{t.d} = tf_{t,d} * idf_t$$

For Example, TI-IDF of AWS = 1/6*0.48 = 0.24 (Based on above calculations)

In a similar way, TF-IDF scores are obtained by multiplying the TF and IDF values of the terms.

In this way, the TF-IDF model is computed, and important words are obtained for further smoother analysis. The output of the TF-IDF Model is further passed as an input to LSI Model.

The TF-IDF model above computes the important words in the corpus, but the context in which the word is being used is not understood. Understanding the context or meaning of the words is as important; hence we have the LSI (Latent Semantic Indexing) model.

- **Latent Semantic Indexing Model (LSI):**

The Latent Semantic Model is an **Unsupervised model** as there is no specific correct or wrong answer for the user query, unlike the supervised model. Algorithms have the freedom to discover and generate interesting, valuable insights from data. Clustering technique is used for semantic search. There is an input from the users for the unsupervised model. The output variable is not present as it learns the underlying structure of the data and provides the most relevant answer to the end-user. Latent Semantic Indexing (LSI) internally uses a **Machine learning model** which is called SVD (Singular Value Decomposition). SVD mainly focuses on Dimension Reduction and performs factorization of a matrix into three matrices.

Exploring the Unsupervised model**, LSI** understands and differentiates the words having similar contextual meanings among the large data sets and is one of the topic modelling algorithms. It identifies the relationship between the context hidden. The actual meaning of LSI is:

Latent – Hidden(evident) but not proved

Semantic – Relationship(context) between words

Indexing – Information retrieval.

Latent Semantic Indexing uses the Singular Value Decomposition (SVD) for reducing the dimensions of the matrix for simple calculations while searching the text with context and avoiding confusion. It creates structured data from unstructured information.

The focus of the LSI model is to map each document and each term to some “concepts”. Here, the concept is nothing but a group of terms with weights assigned.

For Example: AWS Concept: “EC2” (0.8), “RDS” (0.5), “Certificate Manager” (0.4).

A term concept matrix and document concept matrix are prepared from the weights.

Following formula is used for calculating the weight assignments:

$W_{i,j}$ = Occurrences of terms in document

N = Total documents

df_j = Documents containing word

$tf_{i,j}$ = TF-IDF Score

$$W_{i,j} = tf_{ij} * log \frac{N}{df_j}$$

In Latent Semantic Indexing, SVD (Singular Value Decomposition) is used for two purposes:

- Find “concepts” in matrices.
- Dimension reduction or compression g

The next step is to perform the Dimensionality Reduction through SVD (Singular Value Decomposition). SVD is a linear algebra technique that factorizes/decomposes a matrix into three separate matrices products.

The decomposition happens in the following three matrices:

- Orthogonal column matrix (V)
- Orthogonal row matrix (U)
- One Singular Diagonal matrix (S)

Therefore, M= U*S*V

SVD reduces the dimensionality by selecting the largest singular values of the terms t and sustaining only the initial t columns and rows for matrix U and V.

Below is the diagram of the LSI model in which the SVD is zoomed out for detailed working of the algorithm to obtain the output. As shown in the diagram, it indicates that:

Topic Importance (U matrix): Rows are represented as document vectors (in the form of topics).
Topic Distribution across documents (V matrix): Rows represent term vectors (in the form of topics).

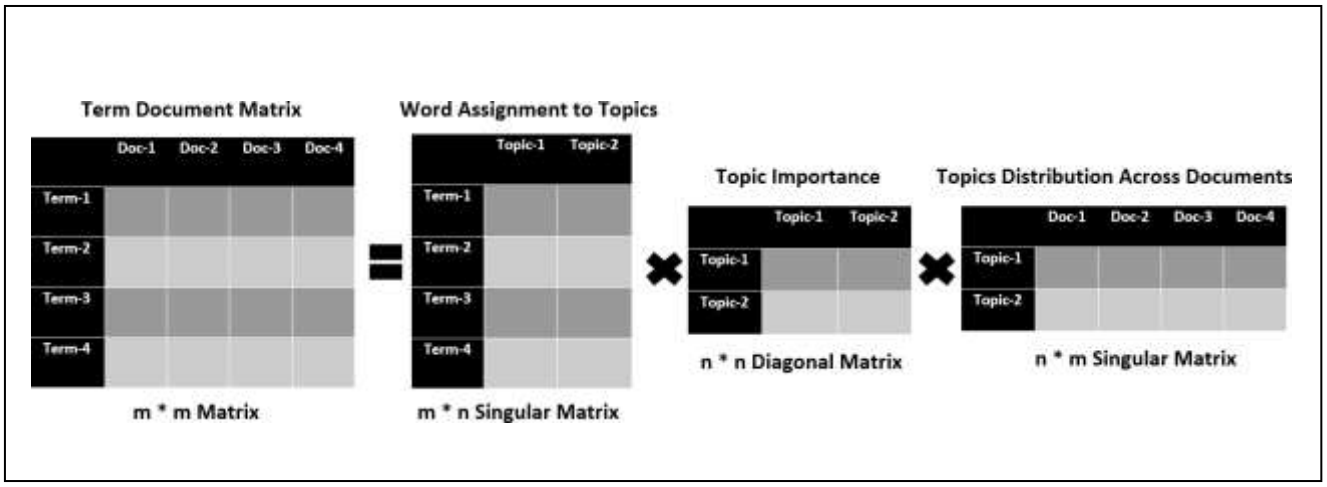
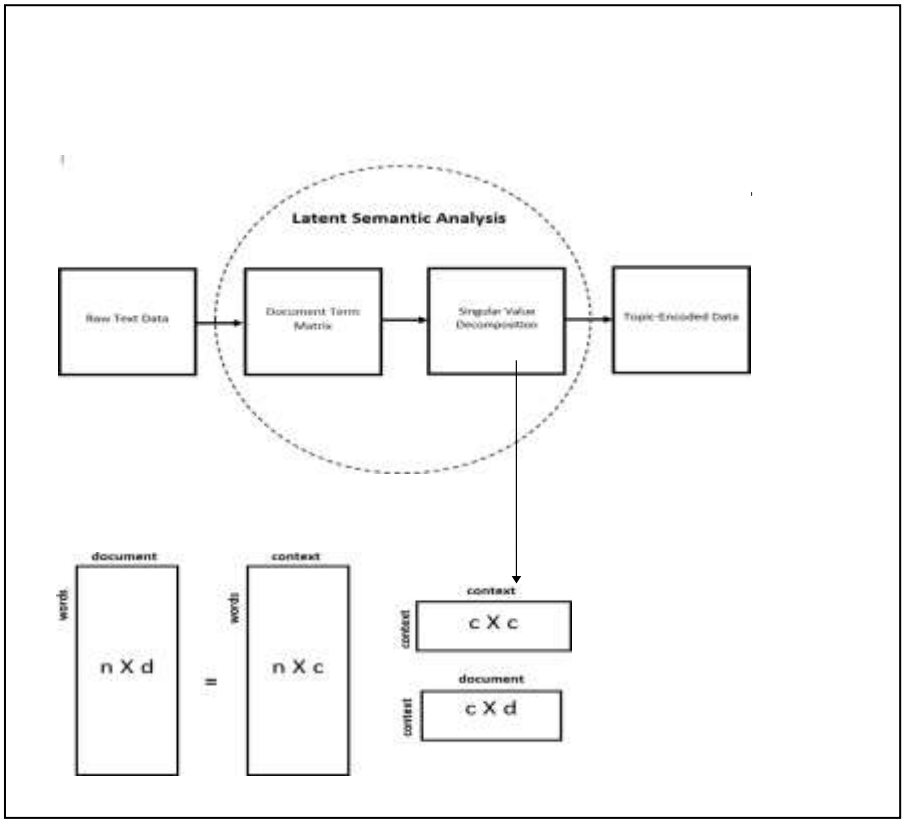


Fig: SVD Matrix Decomposition

Once the SVD performs the reduction and compression operations, every document and term is represented in vector format. And these vectors find the context of similar words and documents with the help of the cosine similarity metric.



Dig: 2

- Hence when a term is searched, it goes through the whole mathematical calculations and processes and matches the context with which the best relevant meaning for that search is retrieved. MatrixSimilarity function provided by Gensim is used particularly for storing the index in the corpus of documents. The unit of measure is cosine between the two vectors.

  Following is the function build to which the actual search query by the user is passed as an input parameter, where the whole flow of calling the algorithms and giving the desired output with the most relevant search percentage is obtained.

1.4 EVALUATION: TECHNOLOGIES DECISION & RESULTS

- **Encoding Techniques:**

ISO – 8859 (Latin1) VS Windows – 11252 (cp1252)

- Encoding must be specified for the text or data from the CSV file here to store the text as binary data. Encoding is nothing but the process of transforming the sequence of code (text data, i.e., dataset CSV file here) to a sequence of bytes (binary data)
- The only difference between the Latin1 and cp1252 encoding is that ISO – 8859 has C1 Control characters, whereas cp1252 has the printable characters.
- Hence, Windows cannot decode byte 0xa0, which is required to read the CSV file dataset and is supported by Latin1.
- Hence "**Latin1**" encoding technique is suitable.

- **Vectorization Techniques:**

Bag of Words VS WordtoVec VS TF-IDF (Term Frequency- Inverse Document Frequency) Model

- Vectorization is carried out because machines do not understand the raw text language directly.

- Hence it needs to be converted to some numerical format which is vectors that are understandable to the machines.
- There are mainly two methods to perform the vectorization –
- **Bag of words** which is the simplest form of text representation in numbers. In which the vocabulary is built just based on the unique occurrences.
- The main drawback of this method is that if a new sentence or word is added, the vocabulary needs to be rebuilt, which indirectly increases the vector size.
- The vector may also consist of many 0's, which leads to a sparse matrix and higher computing cost.
- No sequence of words is maintained in the BoW method.
- In contrast, the TF-IDF Model overcomes all the drawbacks of the BoW method. It calculates the Term Frequency and multiplies it with the Inverse Term Frequency, which gives results not based on text occurrences but also on how important the text is.
- **WordtoVec** encoding technique is a count-based model in which different documents have equal counts for similar words.
- WordtoVec considers the word frequency and converts it to the vector of the frequency of words in the documents.
- WordtoVec only considers the frequency of the words and not the meaning or the context, as the TF-IDF model understands the importance of the word as knowing only the vector with frequency is not sufficient.
- The implementation of WordtoVec below clearly shows that TF-IDF is better in accuracy and understands the important words from the corpus.
- Hence **TF-IDF** is the recommended and suitable method for vectorization from computing cost, accuracy, and better performance than other encoding techniques and considers a result-oriented point of view.

WORDTOVEC IMPLEMENTATION:

Fig: 5.3.2 WordToVec Implementation

- **Topic Modeling Techniques:**

LSI (Latent Semantic Indexing) Model Vs LDA (Latent Dirichlet Allocation) Model

- Latent Dirichlet Allocation is a classification NLP model, while LSI is a count-based model in which different documents have equal counts for similar words. SVD is used internally by LSI to reduce the dimensions.

- LDA tries to perform the cluster by classification of the given problem, which is not suited in the case of semantic search.
- As the data or the text to be searched and understood is huge and cannot be clustered easily.
- If the clusters are created, they would be many clusters as each token would fall under different categories considering the context of the domain.
- The search result is to get the context and retrieve the most relevant search by understating the meaning of the search query, which is very difficult to achieve in clustering and classifying.
- Here the motive of the project is not to predict but to find the most relevant results based on the similarity of data and the understanding context, which the LSI model clearly provides.
- LSI is best suited for finding and analyzing the underlying meaning of documents.
- Below implementation of Latent Dirichlet Allocation (LDA) for the data proves that LSI gives better results and accuracy than LDA for the project scenario.
- Hence the **LSI model** is used for the analysis of the data.

LDA IMPLEMENTATION:

Fig: LDA Implementation

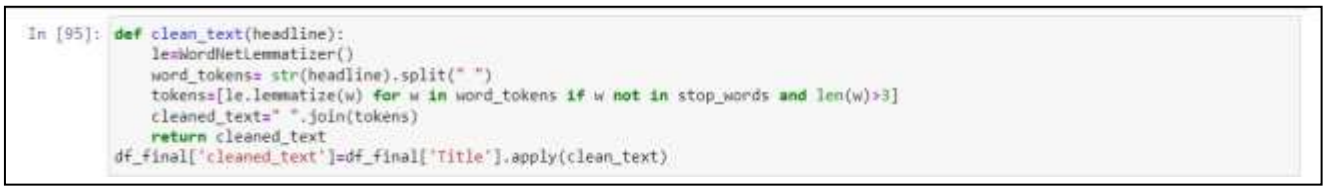


Fig: LDA Implementation

Fig: LDA Implementation

1.4.1. CHATBOT CREATION AND INTEGRATION WITH NLP MODEL

To achieve a user-friendly user interface for showcasing the NLP model results, a chatbot is built in **JavaScript**, which gives the most relevant search result provided by the model at the backend to the user. The front-end UI is written in HTML, CSS, and JavaScript. For the backend, Flask and Python are being utilized here. Below is the integration code between the NLP model and the User Interface.

- **Index File (Backend – CSS, HTML, JavaScript)**

Currently, the chatbot and its implementation are running on the machine's local host. To make it accessible to the users, it requires to be hosted on a server. Cloud is the most used and reliable platform to leverage due to its security and cost optimization features.

- **NLP Chatbot Migration to Aws (Amazon Web Services) Cloud**

Cloud is the on-demand availability of computer resources. Cloud is the new way of optimizing costs by using optimized resources through the "pay as you go" model of the AWS (Amazon Web Services). AWS being a leading cloud provider and automation for migration, the chatbot migrated to the cloud.

AWS resources like EC2 (Elastic Compute Cloud) and S3 (simple storage system) are used. The creation and deployment of these resources in the cloud are performed through automated Terraform templates and Azure DevOps for CICD (Continuous Integration Continuous Deployment) purpose. The user data script has all the shell commands, which will automatically host the application in the cloud. It will also create a new EC2 instance through the Autoscaling group and launch configurations referring to the user data from S3 itself. This whole process automation is achieved and can be deployed within a few minutes, i.e., the application migration to the AWS cloud is completed due to the automation achieved through Terraform, shell scripting and Azure DevOps.

The following resources are created for security, alarming, and deployment purpose through the Terraform templates. EC2 (Elastic Compute Cloud), CloudWatch Alarms, Route53, Autoscaling group, Load Balancer, Security Groups, S3 (Simple Storage Service)

- **Cloud Automation & CICD (Continuous Integration Continues Deployment) Process**

All the resources mentioned earlier (S3, EC2, CloudWatch etc.) of Amazon Web Services (AWS) are deployed through Automation. This automation is highly in demand to avoid manual interruption, saving cost and time for the user. Terraform is one of the most popular Infrastructure as a Code (IAAC) software and can be used by any Cloud provider to deploy the required services like Azure by Microsoft or Google Cloud. Once the Terraform template for the required resources is ready, it is given to Azure Devops. Continuous Integration Continues Delivery is a DevOps practice that provides automation in the building, testing and deployment of applications. AzureDevops is one of the CICD platforms to deploy the resources to the AWS cloud. Azure DevOps connects to the code written for the creation of an infrastructure repository through the Git Repository using the Service connection of the GitHub Repository. Once the connection between GitHub and AzureDevops is established, the pipeline written in YAML, which is present in Git Repository, is executed. The pipeline is used to automatically build and test the project code in the cloud.

- **AWS Resource Creation and Deployment using Terraform.**

  The application is currently running on AWS (Amazon Web Services) cloud with proper resources, a faster, secure, cost-effective, and easily manageable environment.

- **Automation of Application Deployment**

  Once the pipeline in AzureDevops is successfully completed, the application will be up and running in the cloud due to the below user data shell script. This script is automatically executed once the infrastructure is completed in AWS. This script saves time and effort of logging into the EC2 created by the Terraform template above and running the steps one by one manually to make the application up and running.

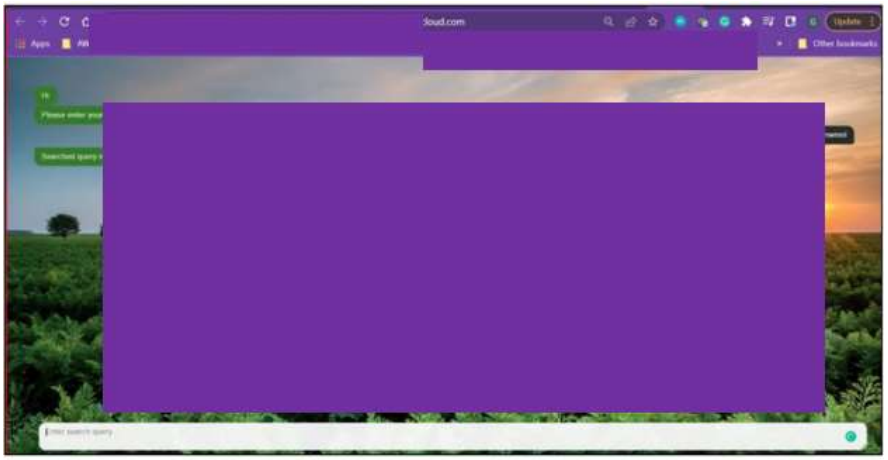
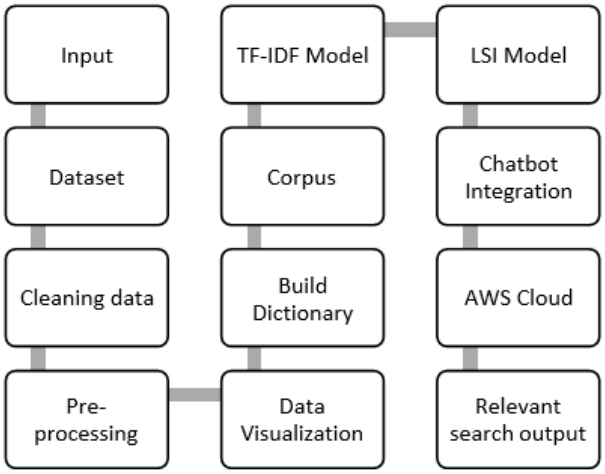


Fig: Chatbot hosted on AWS

1.4.2 Process Flow



Dig: 3

1.5 CONCLUSION & FUTURE OF WORK

- **Conclusion**

The work accomplished in this project solves the impediments of the users, the Natural Language Processing Chatbot learns the previous already solved issues from years accurately and gives the users the best relevant results to the query asked by the users within seconds. This saves users time and manual efforts along with a user-friendly interactive chatbot. Hosting the application on the cloud has the advantage of high availability, security, and fast accessibility for the users. Comparative studies of Natural Language Processing algorithms, namely LSI and LDA, are present. However, for semantic search, LSI models have better accuracy than LDA. At the same time, the different encoding techniques studies showed that TF-IDF is the best-resulting giving algorithm due to its log formula compared to WordtoVec and BagofWords model. This automation enables us to achieve significant results in time cost reduction.

- **Future Of Work**

The current model can be enhanced by using the sophisticated advanced deep learning model of Google's called BERT (Bidirectional Encoder Representations from Transformers) Deep neural network AI technology, which would add voice command features like Siri and Alexa. This provides users easy access to the application from everywhere and anytime.

REFERENCES

[1] S. V. M. Y. Pandiarajan, "Semantic Search Engine Using Natural Language Processing," Advanced Computer and Communication Engineering Technology. Springer, Cham, pp. 561-571, 2015.

[2] A. R. S. S. A. Kupiyalova, "Semantic search using Natural Language Processing," 2020 IEEE 22nd Conference on Business Informatics (CBI), vol. 2, pp. 96-100, 2020.

[3] N. J. Belkin, "Anomalous states of knowledge as a basis for information retrieval," Canadian journal of information science, pp. 133-143, 1980.

[4] K. M. Yri, "Cliff Goddard. Semantic analysis. A practical introduction," Functions of Language, pp. 127-137, 2014.

[5] W. F. J. L.-P. P. R. M.-M. J. N.-I. a. F. J. Z.-S. Renteria-Agualimpia, "Exploring the advances in semantic search engines," Distributed Computing and Artificial Intelligence, pp. 613-620, 2010.

[6] S. T. G. W. F. T. K. L. S. D. a. R. H. Dumais, "Using latent semantic analysis to access to textual information" Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 281-285, 1988.

[7] H. S. A. M. A. a. K. M. M. Alatawi, "Detecting white supremacist hate speech using domain specific word embedding with deep learning and BERT," IEEE Access, pp. 106363-106374, 2021.

[8] T. Doszkocs, "The Future of Semantic Search," [Online]. Available: https://citeseerx.ist.psu.edu/