Eastern University
A Leader in quality Education

Thesis Paper

A Thesis on "**Corpus Analyzer – Machine Learning and Deep Learning Based Virtual Assistant**"

Submitted by:

Mohammad Abdullah Al Salman

Student ID: 183400026

Md. Reaj Uddin

Student ID: 183400043

Jahanara Begum

Student ID: 181400026

Md. Mostofa Jalal

Student ID: 183400042

Md. Rifat Hasan

Student ID: 182400056

Emdad Hossain

Student ID: 181400102

In fulfillment of the requirements for the degree of Bachelor Science in Computer Science and Engineering

CANDIDATE'S DECLARATION

This is hereby declared that the work titled "**Corpus Analyzer – Machine Learning and Deep Learning Based Virtual Assistant**" is the outcome of research carried out by us under the supervision of Prof. Dr. Md. Mahfuzur Rahman, in the Department of Computer Science and Engineering, Eastern University. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Mohammad Abdullah Al Salman

Md. Reaj Uddin

Jahanara Begum

Md. Mostofa Jalal

Md. Rifat Hasan

Emdad Hossain

APPROVAL

This project titled **"Corpus Analyzer – Machine Learning and Deep Learning Based Virtual Assistant**" submitted by Mohammad Abdullah Al Salman (ID: 181400026), Md. Reaj Uddin (ID: 183400043), Jahanara Begum (ID: 181400026), Md. Mostofa Jalal (ID: 183400042), Md. Rifat Hasan (ID: 182400056), Emdad Hossain (ID: 1814000102) has been accepted satisfactory in partial fulfillment of the requirement for the degree Bachelor of Science in Computer Science and Engineering.

Board of Examiners

ACKNOWLEDGEMENT

Firstly, all praise to the Great Almighty for whom our thesis have been completed without any major interruption.

This thesis has been supervised by Prof. Dr. Md. Mahfuzur Rahman Dean, Faculty of Engineering and Technology, & Chairperson, Dept. of Computer Science and Engineering, Eastern University, Dhaka, Bangladesh. We are very much grateful and indebted to him for his kind suggestion, guidance, instructions and overall supervision that he offered us during our thesis investigation. We also thank him for reviewing the preliminary versions of this thesis and making useful corrections as well as providing valuable suggestions.

We would also like to express our gratitude to our parents and teachers for bringing us up where we are today. We are thankful to the Department of Computer Science & Engineering, Eastern University, for providing us with an excellent educational environment and computing facility. We also offer our best regards Eastern University administration, and all other esteemed teachers of the department for their affectionate feelings and encouragement throughout the period of our research work.

Finally, warm thanks to all of our well-wishers and friends for moral support and inspiration.

ABSTRACT

The modern world is progressing quickly along with technology and one of the major sectors is Artificial Intelligence. Artificial Intelligence enhances the speed, precision and effectiveness of human efforts. People use artificial intelligence in many sectors. One of the sectors is assistant. In the market lot's of artificial intelligence assistance is available. Some are google assistant, alexa and siri. They all are artificial intelligence based virtual assistants. They give answers to the questions that people ask and also do other stuff. When they answer the question, they do not follow the context of the question. Our thesis is based on this issue. We make such artificial intelligence that follows the context of the question and then it will answer that question. Our thesis is inspired by the above virtual assistants. We make such artificial intelligence that makes us answer based on our discussion context. And also it helps us to find the summary of the discussion and find the best article on the web. And it also has a self learn capability feature. And we are very glad to say that, we build such artificial intelligence that gives us the answer based on our discussion context. And we are very happy with the result of our artificial intelligence.

Table of Content

Chapter One

Introduction

1.1 Objective

We make such an artificial intelligence that makes our life easier. Our artificial intelligence has below features.

- Determine the discussion topic and give the question's answer based on discussion context.
- Give answers from any given topic.
- Find the best article based on the search query in the offline dataset.
- Give a summary for any article.
- Find the best article from online based on a search query.
- Learn about the unknown topic through online and give question's answer about that topic.
- And multi language features.

1.2 Motivation

None of the services from the major tech companies was close to perfect. If I ask Google Assistant "What is DC meaning?", It says about Direct Current but I need to know about Debit Cards. When I asked Alexa earlier "This week who was playing in the Super Bowl", she responded, somewhat monotonously, "Super Bowl 49's winner is New England Patriots." But the match was played some year before.

Search Engines like Google, Yahoo, Bing, etc. stay in business by selling ads with access to large audiences. The product they sell is not search results but rather access to searchers. With this in mind, the goal of every Search Engine is to grow their search audience, expanding their product offering to companies willing to pay for more ads. They also understand that searchers must trust them to provide the results they are looking for, for them to come back and search again.

But we want to provide vetted, trusted and valuable content to our searchers. We want to provide the exact thing that they want, not any irrelevant thing. The original idea behind us was to allow users to get answers to questions posed in everyday, natural language, as well as by traditional keyword searching.

1.3 Technology Used

- **Language**
  - Python
  - HTML
  - CSS
  - JavaScript

- **Framework**
  - Flask
  - Selenium

- **Libraries**
  - Scikit-learn
  - Pickle
  - Time
  - OS
  - AllenNLP
  - Newspaper3k
  - PyTorch
  - NLTK(Natural Language Toolkit)

- **API**
  - Free Google Translate API for Python
  - Free Google Search API for Python
  - ngrok (Local port to public url)

Chapter Two

Problem Statement

2.1 Problem Statement

There are lots of virtual assistants available. Those help people to get answers. But they do not follow the question's context. Our main goal of this thesis is to make such an artificial intelligence that gives answers based on context’s discussion.

Chapter Three

Literature Review

3.1 Previous Related Work

This section includes the work already done on this thesis by various researchers using different methodologies and algorithms. Following is the brief description of them:

3.1.1 BERT

Pre-training of Deep Bidirectional Transformers for Language Understanding. *[Submitted on 11 Oct 2018 (v1), last revised 24 May 2019 (this version, v2)]*

In paper they introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

They have shown that the standard BERT recipe (including model architecture and training objective) is effective on a wide range of model sizes, beyond BERT-Base and BERT-Large. The smaller BERT models are intended for environments with restricted computational

resources. They can be fine-tuned in the same manner as the original BERT models. However, they are most effective in the context of knowledge distillation, where the fine-tuning labels are produced by a larger and more accurate teacher.

BERT is a method of pre-training language representations, meaning that we train a general-purpose "language understanding" model on a large text corpus (like Wikipedia), and then use that model for downstream NLP tasks that we care about (like question answering). BERT outperforms previous methods because it is the first unsupervised, deeply bidirectional system for pre-training NLP.

Unsupervised means that BERT was trained using only a plain text corpus, which is important because an enormous amount of plain text data is publicly available on the web in many languages.

Pre-trained representations can also either be context-free or contextual, and contextual representations can further be unidirectional or bidirectional. Context-free models such as word2vec or GloVe generate a single "word embedding" representation for each word in the vocabulary, so banks would have the same representation in bank deposit and river bank. Contextual models instead generate a representation of each word that is based on the other words in the sentence.

BERT was built upon recent work in pre-training contextual representations — including Semi-supervised Sequence Learning, Generative Pre-Training, ELMo, and ULMFit — but crucially these models are all unidirectional or shallowly bidirectional. This means that each word is only contextualized using the words to its left (or right). For example, in the sentence I made a bank deposit the unidirectional representation of the bank is only based on I made a but not deposit.

Some previous work does combine the representations from separate left-context and right-context models, but only in a "shallow" manner. BERT represents "bank" using both its left and right context — I made a ... deposit — starting from the very bottom of a deep neural network, so it is deeply bidirectional.

3.1.2 BiDAF

Bidirectional Attention Flow for Machine Comprehension.*[Submitted on 5 Nov 2016 (v1), last revised 21 Jun 2018 (this version, v6)]*

Machine comprehension (MC), answering a query about a given context paragraph, requires modeling complex interactions between the context and the query. Recently, attention mechanisms have been successfully extended to MC. Typically these methods use attention to focus on a small portion of the context and summarize it with a fixed-size vector, couple attention temporally, and/or often form a uni-directional attention. In this paper we introduce the Bi-Directional Attention Flow (BIDAF) network, a multi-stage hierarchical process that represents the context at different levels of granularity and uses bi-directional attention flow mechanism to obtain a query-aware context representation without early summarization. Our experimental evaluations show that our model achieves the state-of-the-art results in Stanford Question Answering Dataset (SQuAD) and CNN/DailyMail cloze test.

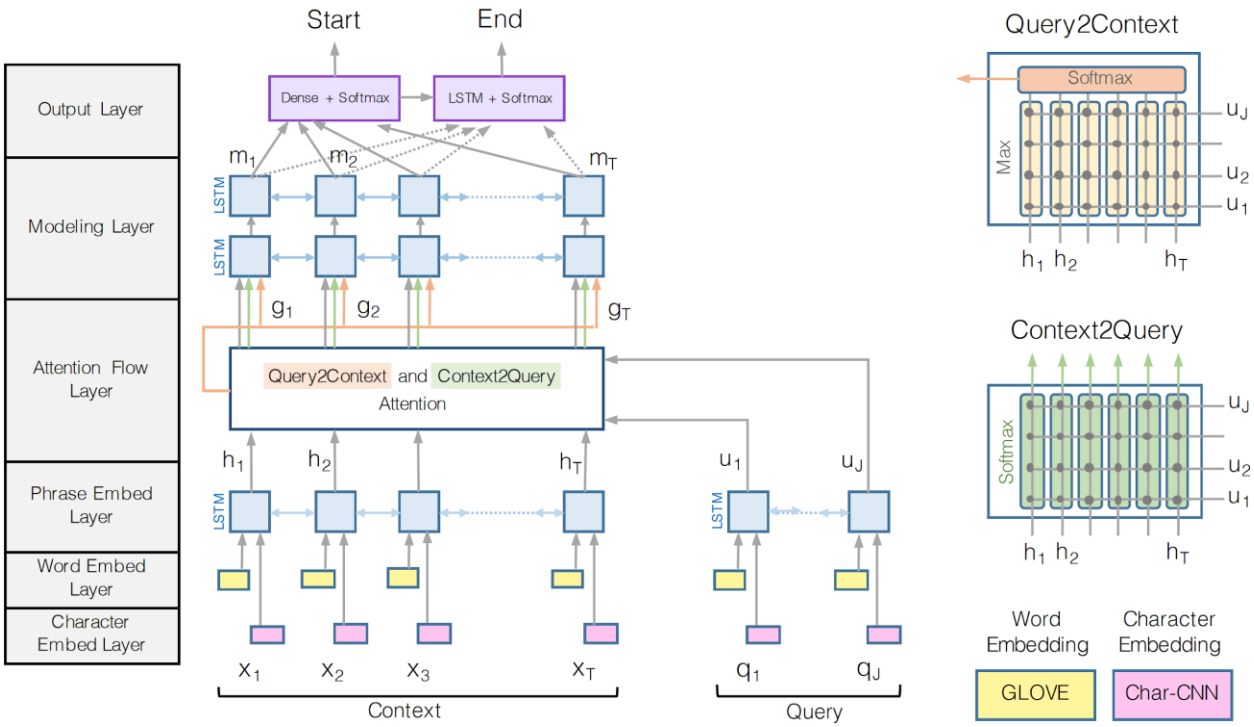


Figure: Bi-Directional Attention Flow for Machine Comprehension

Chapter Four

Frameworks, Libraries and API

For our thesis we use three languages, two frameworks, eight libraries and three api. The languages are Python, HTML, CSS, JavaScript. The frameworks are Flask and Selenium. The libraries are scikit-learn, pickle, time, os, AllenNLP, Newspaper3k, PyTorch, nltk(Natural Language Toolkit). And API's are Google Translate API, Google Search API and ngrok API.

4.1 Language

- **Python**

  Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Python has a standard library in development, and a few for AI. It has an intuitive syntax, basic control flow, and data structures. It also supports interpretive run-time, without standard compiler languages. This makes Python especially useful for prototyping algorithms for AI.

- **HTML**

  Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript and VBScript.

- **CSS**

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

- **JavaScript**

JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

It’s an object-oriented computer programming language commonly used to create interactive effects within web browsers.

4.2 Framework

- **Flask**

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

- **Selenium**

Selenium is a portable framework for testing web applications. Selenium provides a playback tool for authoring functional tests without the need to learn a test scripting language.

Selenium is an open-source web-based automation tool. Python language is used with Selenium for testing. It has far less verbose and easy to use than any other programming language. Selenium can send the standard Python commands to different browsers, despite variation in their browser's design.

4.3 Libraries

- **Scikit-learn**

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

- **Pickle**

  Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.)

- **Time**

  The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of any code.

- **OS**

  The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

- **AllenNLP**

  An open-source NLP research library, built on PyTorch. AllenNLP makes it easy to design and evaluate new deep learning models for nearly any NLP problem, along with the infrastructure to easily run them in the cloud or on your laptop. AllenNLP includes reference implementations of high quality models for both core NLP problems (e.g. semantic role labeling) and NLP applications (e.g. textual entailment).

- **Newspaper3k**

  Newspaper is a Python3 library. Article scraping & curation. Inspired by requests for its simplicity and powered by lxml for its speed. Newspaper is an amazing python library for extracting & curating articles. Newspaper delivers Instapaper style article extraction. It is a very fast and easy way to scrape articles from url. It uses a python request library.

- **PyTorch**

  PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab. It is free and open-source software released under the Modified BSD license. PyTorch is a library for Python programs that facilitates building deep learning projects.

- **NLTK(Natural Language Toolkit)**

  The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

4.4 API

- **Google Translate API**

  Googletrans is a free and unlimited python library that implemented Google Translate API. This uses the Google Translate Ajax API to make calls to such methods as detect and translate. Compatible with Python 3.6+. It has this feature. It is Fast and reliable. It uses the same servers that translate.google.com uses. Auto language detection. Bulk translations. Customizable service URL. HTTP/2 support.

- **Google Search API**

  googlesearch is a Python library for searching Google, easily. googlesearch uses requests and BeautifulSoup4 to scrape Google. It's free and very fast. It's a third party library.

- **ngrok API**

  pyngrok is a Python wrapper for ngrok that manages its own binary and puts it on your path, making ngrok readily available from anywhere on the command line and via a convenient Python API.

  ngrok is a reverse proxy tool that opens secure tunnels from public URLs to localhost, perfect for exposing local web servers, building webhook integrations, enabling SSH access, testing chatbots, demoing from your own machine, and more, and its made even more powerful with native Python integration through pyngrok.

Chapter Five

Implementation

The whole process can be divided into 6 major parts:-

1. Data Collection
2. Categorize Data
3. Convert to tf–idf
4. Apply ML for all category
5. Apply ML for sub category
6. Send data in BiDAF model

5.1 Data Collection

We collect data from google search engine. We are collecting data on a total of nine categories. The categories are:

1. Animation and Comics
2. Computers
3. Education
4. Food and Drink
5. Health
6. Jobs and Career
7. Pets and Animals
8. Politics
9. Religion

For collecting data we use selenium framework. Selenium is an open-source web-based automation tool. Python language is used with Selenium for testing. It has far less verbose and

easy to use than any other programming language. Selenium can send the standard Python commands to different browsers, despite variation in their browser's design.

We collect 1794 articles for the Animation and Comics category. Which has a total 2504098 words and 12742886 characters.

We collect 4891 articles for the Computers category. Which has a total 1255386 words and 7230500 characters.

We collect 4052 articles for the Education category. Which has a total 3127654 words and 17300185 characters.

We collect 4635 articles for the Food and Drink category. Which has a total 3121518 words and 15808709 characters.

We collect 4087 articles for the Health category. Which has a total of 3064762 words and 16554156 characters.

We collect 4875 articles for the Jobs and Career category. Which has a total 2871646 words and 15043733 characters.

We collect 4375 articles for the Pets and Animals category. Which has a total 3669088 words and 18162023 characters.

We collect 2625 articles for the Politics category. Which has a total 1215305 words and 5805350 characters.

We collect 2424 articles for the Religion category. Which has a total 934206 words and 4471838 characters.

In total we collect 33758 articles which contains total 21763663 words and total 113119380 characters .

Our data collection approach was below,

First we search the category in google keyword planner. Google keyword planner gives us keyword ideas about that category. For example when we search “Health” in google keyword planner, they gives below keywords:

health, apple cider vinegar, smoking, optometrist, healthy snacks,nutrition, healthy food near me, healthy food, disease, health department, hipaa, covered california, epidemiology, healthy, healthy breakfast, nurse practitioner, clinic, junk food, tricare online, world health organization, health equity, healthy meals, healthcare, green tea benefits, mental etc.

Google keyword planner gives us a total 140 related keywords for the health category.

Then we search every keyword in google search engine and collect articles from them.

Then we found a total 4087 articles for the health category.

We repeat the above process for all nine categories. So this is a repeated task. Every time we search a keyword in google search engine and then collect articles from google search results. That’s why we take help from the selenium framework and Newspaper3k librarie. Selenium framework collects links from google search results then Newspaper3k collects articles from that links. In this way we collect 4087 articles for the health category.

We repeat this process for all our nine categories. So it makes our work fast.

5.2 Categorize Data

So now we have a total of 33758 articles. Now we will put each article in its own category. We use python list data structure for holding our articles. We have a totally nice giant list whose individual list has lots of articles. Then we convert that giant list to pickle file. Because we have to use this list lots of time. Pickle file converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. So we can use this pickle file easily anywhere.

5.3 Convert to tf–idf

So now we have lots of articles. Which all are string. We all know, computers can't understand strings. It only understands numbers. So now our challenge is to convert those articles to meaningful numbers.

There are lots of techniques available for converting strings to meaningful numbers. Some are Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF) and Word2vec technique.

We use the TF-IDF technique for our thesis. TF-IDF short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

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

Here, in the numerator, n is the number of times the term "t" appears in the document "d". Thus, each document and term would have its own TF value. For better understanding see the below example.

Suppose we have three articles.

Review 1: This movie is very scary and long

Review 2: This movie is not scary and is slow

Review 3: This movie is spooky and good

We will first build a vocabulary from all the unique words in the above three reviews. The vocabulary consists of these 11 words: ‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’.

For finding TF, let’s consider Review 2

Number of words in Review 2 = 8

TF for the word ‘this’ = (number of times ‘this’ appears in review 2)/(number of terms in review 2) =1/8

Similarly,

TF(‘movie’) = 1/8

TF(‘is’) = 2/8 = 1/4

TF(‘very’) = 0/8 = 0

TF(‘scary’) = 1/8

TF(‘and’) = 1/8

TF(‘long’) = 0/8 = 0

TF(‘not’) = 1/8

TF(‘slow’) = 1/8

TF(‘spooky’) = 0/8 = 0

TF(‘good’) = 0/8 = 0

We can calculate the term frequencies for all the terms and all the reviews in this manner:

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

So we have to find IDF.

IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words:

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

We can calculate the IDF values for the all the words in Review 2:

IDF('this') = log(number of documents/number of documents containing the word 'this') = log(3/3) = log(1) = 0

Similarly,

IDF('movie',) = log(3/3) = 0

IDF('is') = log(3/3) = 0

IDF('not') = log(3/1) = log(3) = 0.48

IDF('scary') = log(3/2) = 0.18

IDF('and') = log(3/3) = 0

IDF('slow') = log(3/1) = 0.48

We can calculate the IDF values for each word like this. Thus, the IDF values for the entire vocabulary would be:

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

Hence, we see that words like “is”, “this”, “and”, etc., are reduced to 0 and have little importance; while words like “scary”, “long”, “good”, etc. are words with more importance and thus have a higher value.

We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

We can now calculate the TF-IDF score for every word in Review 2:

TF-IDF(‘this’, Review 2) = TF(‘this’, Review 2) * IDF(‘this’) = 1/8 * 0 = 0

Similarly,

TF-IDF('movie', Review 2) = 1/8 * 0 = 0

TF-IDF('is', Review 2) = 1/4 * 0 = 0

TF-IDF('not', Review 2) = 1/8 * 0.48 = 0.06

TF-IDF('scary', Review 2) = 1/8 * 0.18 = 0.023

TF-IDF('and', Review 2) = 1/8 * 0 = 0

TF-IDF('slow', Review 2) = 1/8 * 0.48 = 0.06

Similarly, we can calculate the TF-IDF scores for all the words with respect to all the reviews:

Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

We have now obtained the TF-IDF scores for our vocabulary. TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e the word is rare in all the documents combined but frequent in a single document.

In this above way we convert our all articles to Term Frequency-Inverse Document Frequency (TF-IDF).

So this technique works completely fine for our thesis. We didn't use the Bag-of-Words technique because Bag-of-Words has some drawbacks for our thesis. The drawbacks are:

- If the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase too.
- Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix (which is what we would like to avoid)
- We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text.

5.4 Apply ML for all category

Now we have a huge articles list and we converted that articles to numbers. Now it's time to apply machine learning algorithms. There are lots of machine learning algorithms available. Like Linear Regression, Logistic Regression, Decision Tree, SVM, Naive Bayes,k NN, K-Means, Random Forest, Dimensionality Reduction Algorithms, Gradient Boosting algorithms, GBM,XGBoost, LightGBM, CatBoost etc.

For our dataset we use Naive Bayes MultinomialNB algorithm. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of particular words, multinomial naive Bayes explicitly models the word counts and adjusts the underlying calculations to deal with it.

Now we apply the MultinomialNB algorithm for our dataset. In Python there is a library called scikit-learn. Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features

various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and MultinomialNB.

So we just use this scikit-learn library to apply the MultinomialNB algorithm.

Our input was all nine giant lists. And output was integer 0 to 8. 0 means first category, 1 means second category and so on.

In this way we build such artificial intelligence that takes input and gives output about any nine categories. This artificial intelligence only can say, given string is which category. It can't detect the actual article where the given string belongs to. Our next section solves this issue.

5.5 Apply ML for sub category

In the above section we successfully determined the given string is which category. Now our task is to find the exact article where the given string belongs to.

Now again we apply Naive Bayes MultinomialNB algorithm. Because we are working with text. But this time our goal is to find the exact article where our article belongs to. That's why our input and output will be different.

We apply the MultinomialNB algorithm nine times for individual lists. For example if our category is health then our input will be all the 4087 articles. And our output will be 0 to 4086. 0 means first article, 1 means second article and so on.

We apply this process for all our nine categories. In this way our AI can determine which articles a given string belong to. We apply all the algorithms using scikit-learn library.

From the above section our AI determines the given string is which category and after finding the category it finds the article in that category where given input belongs to. In this way we get the exact article from our huge article lists.

5.6 Send data in BiDAF model

Now we have the exact article where given input belongs. Now we feed this article to BiDAF model. BiDAF is a pre-trained deep learning model by AllenNLP. This model is also called Bi-Directional Attention Flow for Machine Comprehension. This model is trained by SQuAD2.0 dataset. SQuAD2.0 is a Stanford Question Answering Dataset.

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.

SQuAD2.0 combines the 100,000 questions in SQuAD1.1 with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. To do well on SQuAD2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

This model is capable of answering the question from the given article. So our task is just to send the article in this model with a question. This model gives us the possible answer for that question.

So we just send the article which we found from the above section with a question. The model will give us the answer.

So finally we found such an answer that belongs to our context.

Chapter Six

Results

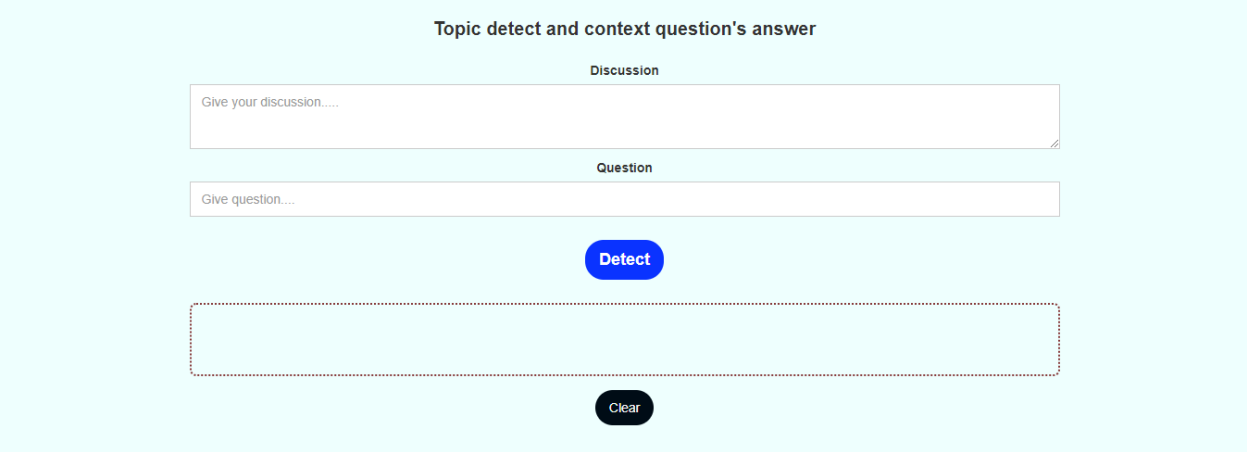
We implement our whole thesis into a website. So anyone can use that website if the website is live. We make our website using the Flask framework. Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

We have divided our AI into six sections. It’s section has a unique feature.

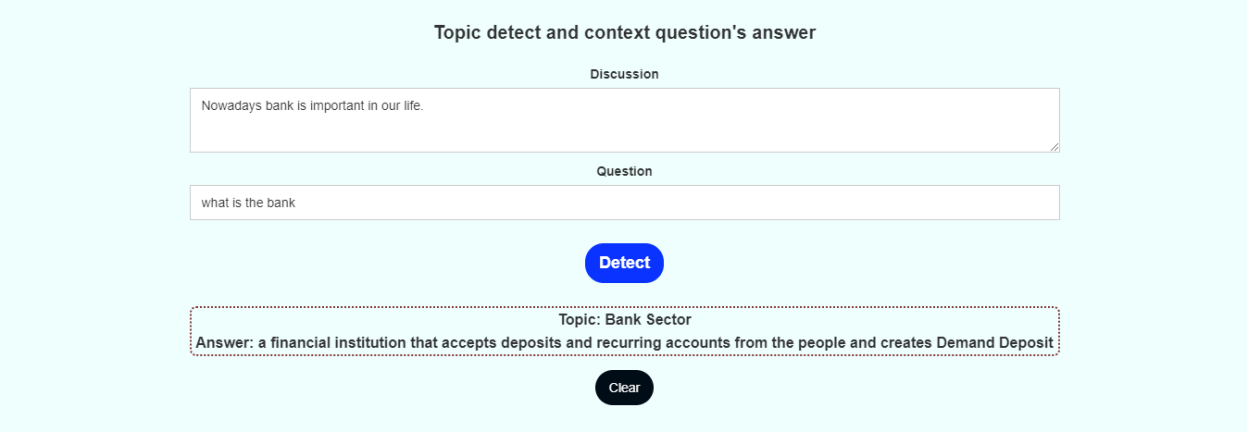
- ➔ Section 1: Topic detect and context question's answer
- ➔ Section 2: Any article and answer from that article
- ➔ Section 3: Search article in offline
- ➔ Section 4: Get summary from the article.
- ➔ Section 5: Search online and get related article
- ➔ Section 6: AI learn by its own

6.1 Topic detect and context question's answer

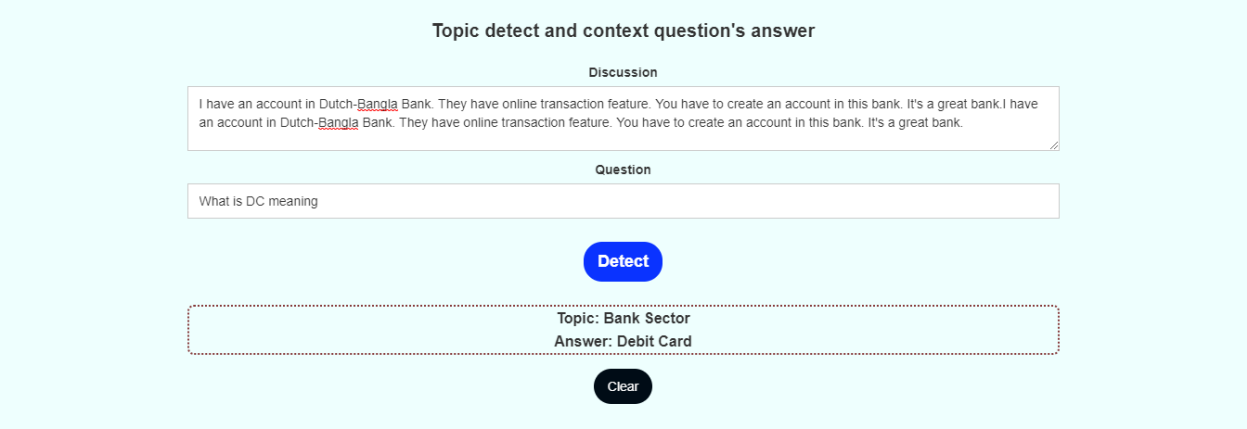
In this section our artificial intelligence gives an answer based on the discussion context. Here our AI needs two inputs. First is discussion and second is question. In the website our first section looks like below picture.



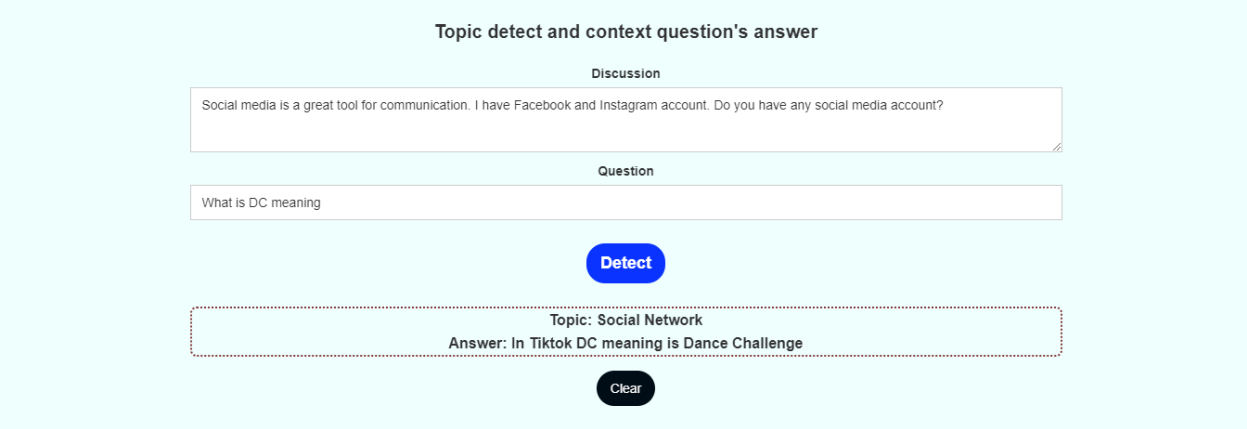
People just give their discussion in the discussion box and ask a question in the question box. After giving the discussion and asking questions just press the Detect button. Then our AI will give the answer.



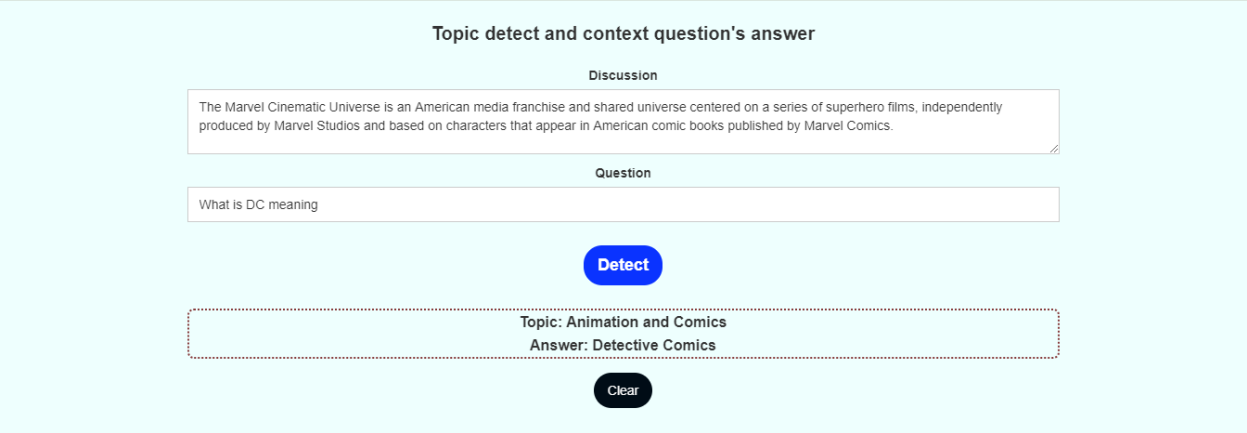
Here our AI determines two things. First is what topics are discussed and second is what is the answer of the question based on discussion context. Given I give some other examples, that can help to better understand.



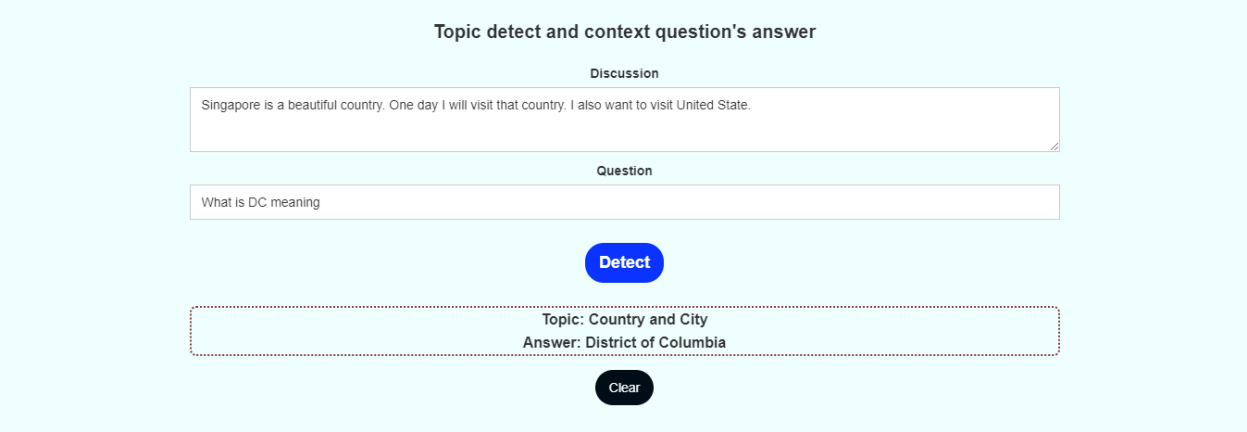
In the above picture our question is “what is DC meaning”. Our AI answer that question is Debit Card. Because above our discussion topic is bank related. That’s why AI gives bank related answer.



Again our question is the same. “what is DC meaning”. But this time our AI answer is different. The AI gives DC meaning is Dance Challenge.

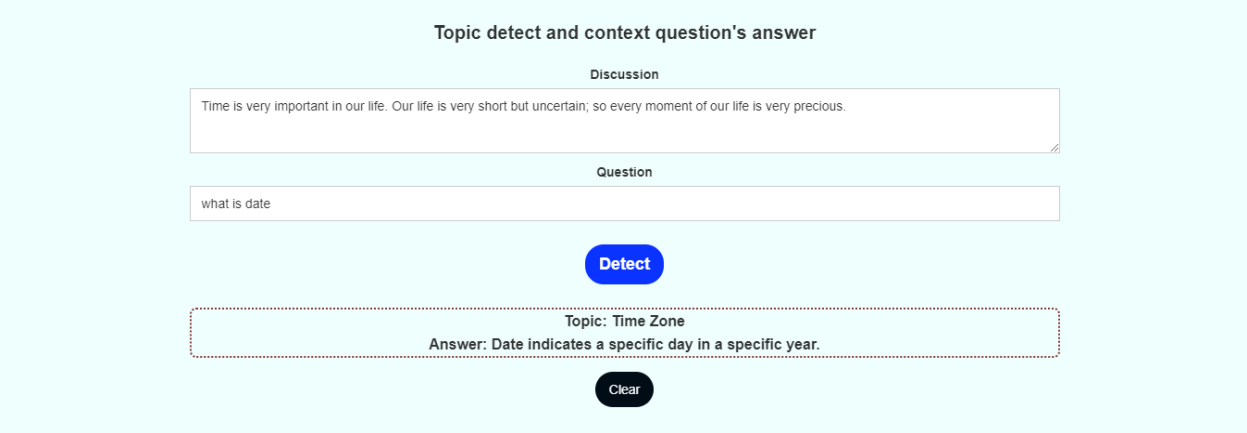


This time our AI gives a different answer for the same question.

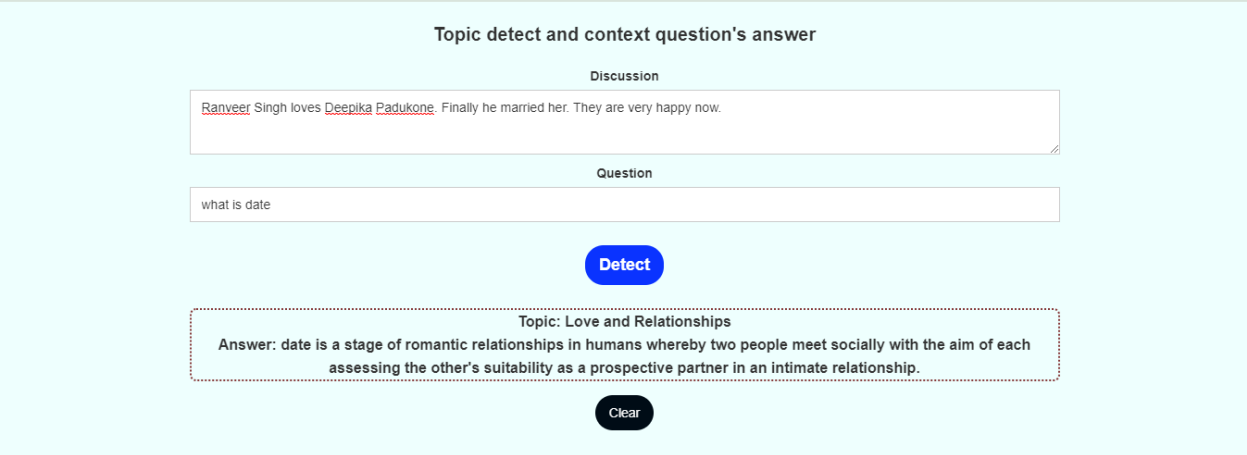


This time the AI answer is District of Columbia. Because it detects discussion topics are Country and city. That's why AI detect DC means Washington, D.C in the United State of America.

See some other examples.

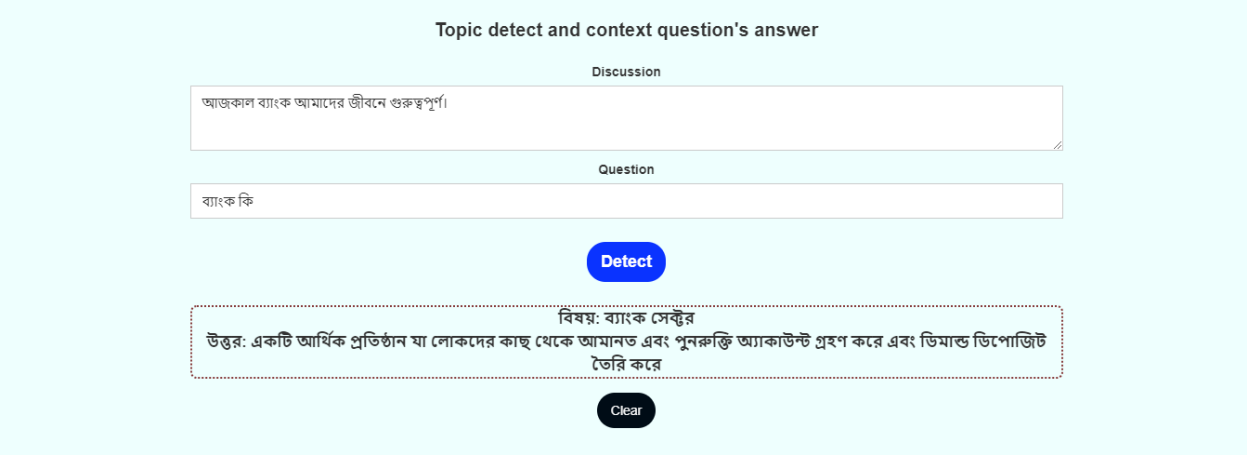


Our question is what is date. See the answer.



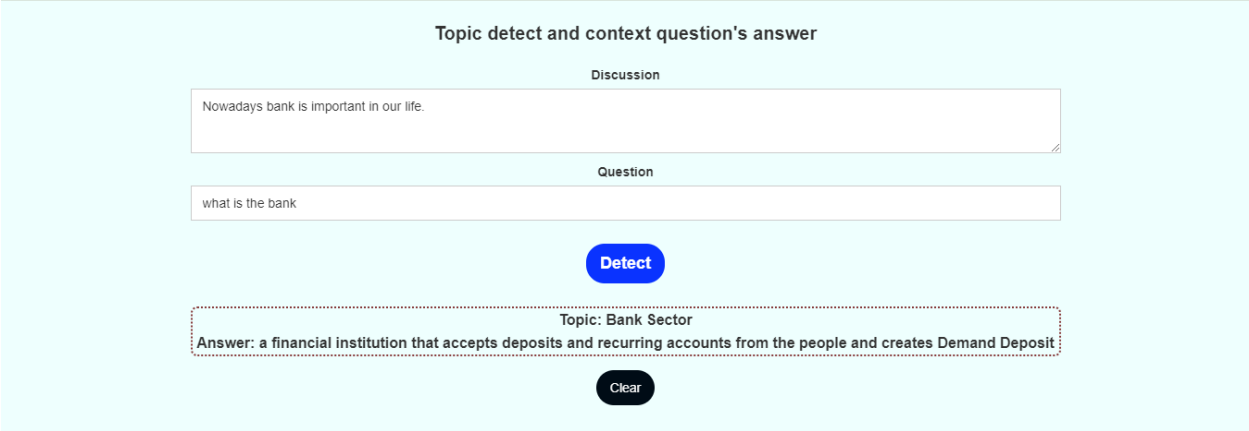
Now again our question is what is date. Now see the answer. The answer is different. Because our context is different.

In this first section it has multi language features available. Below are given some examples.

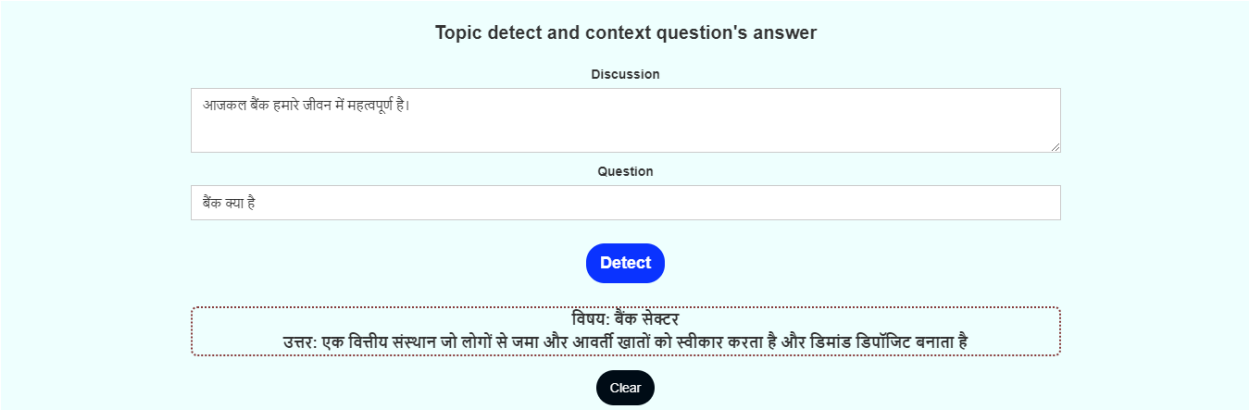


Detect Bangla Language.

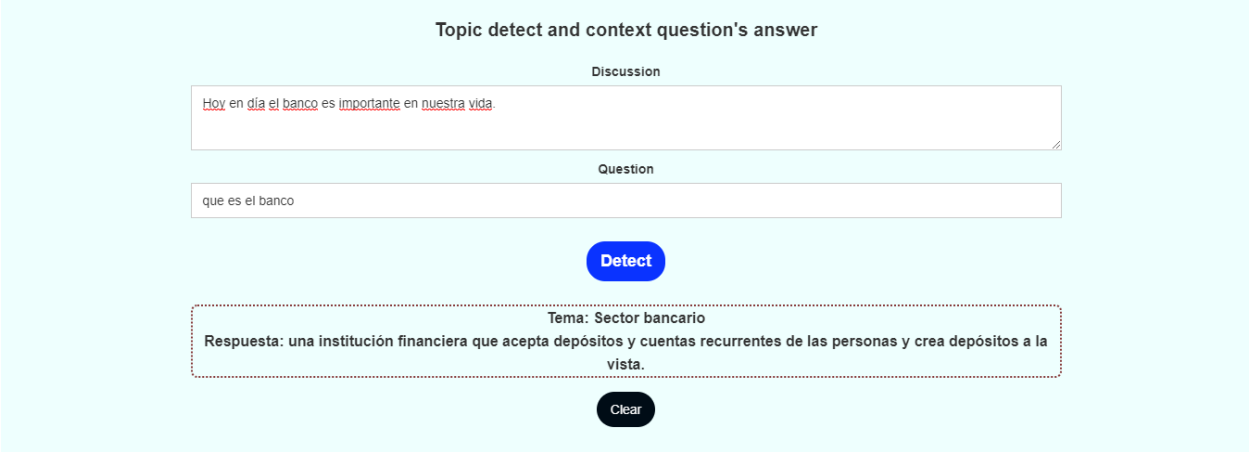
Detect Chinese Language



Detect English Language



Detect HindiLanguage



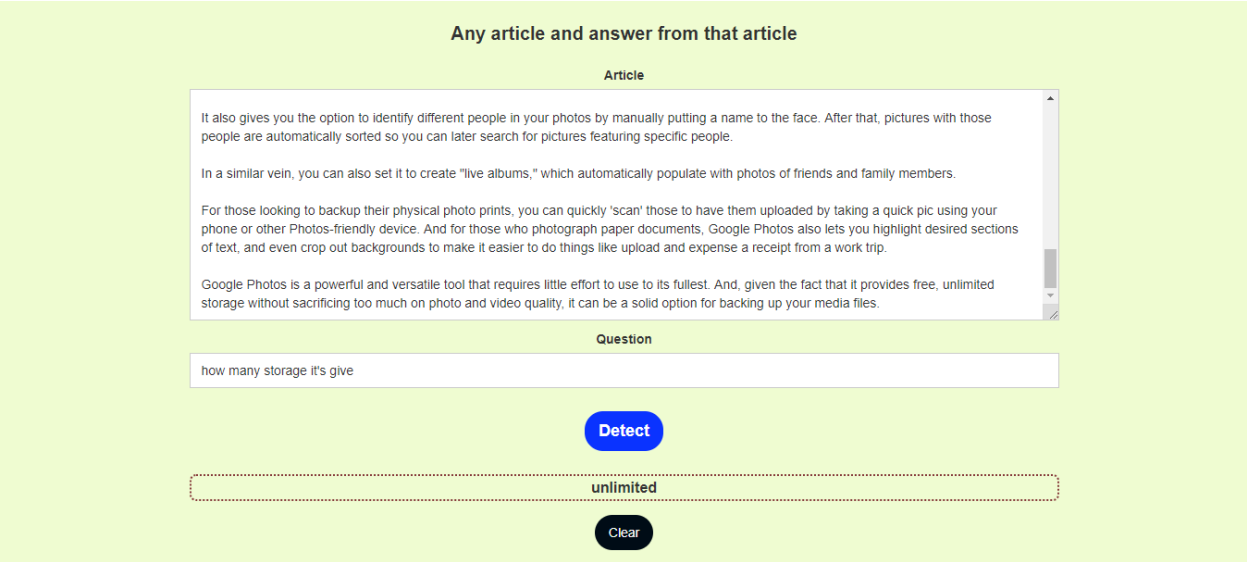
Detect SpanishLanguage

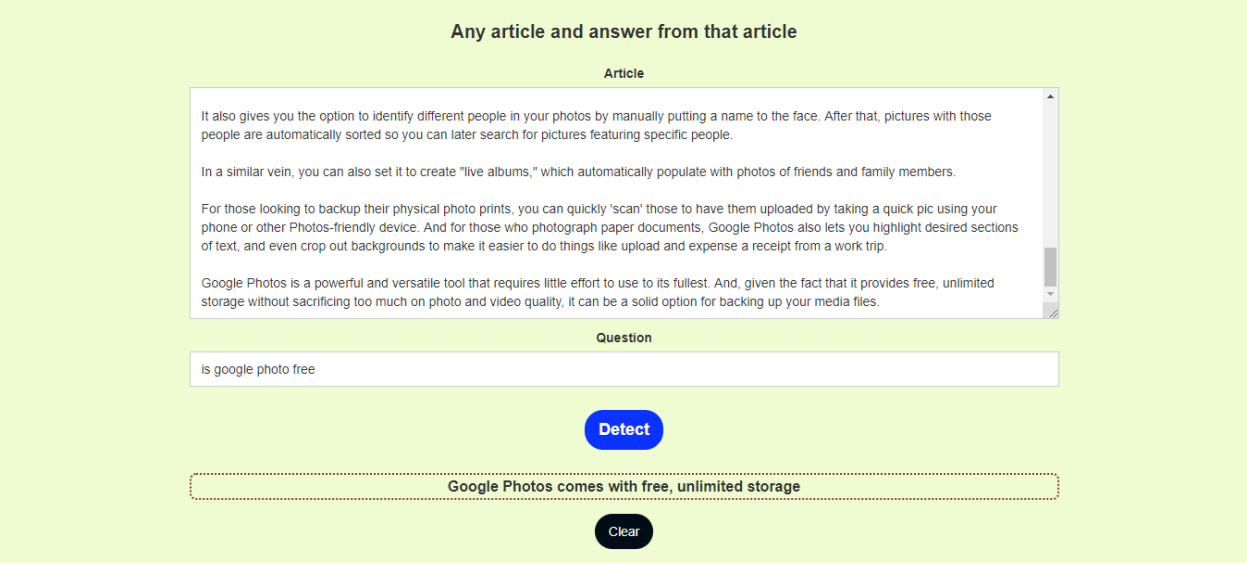
6.2 Any article and answer from that article

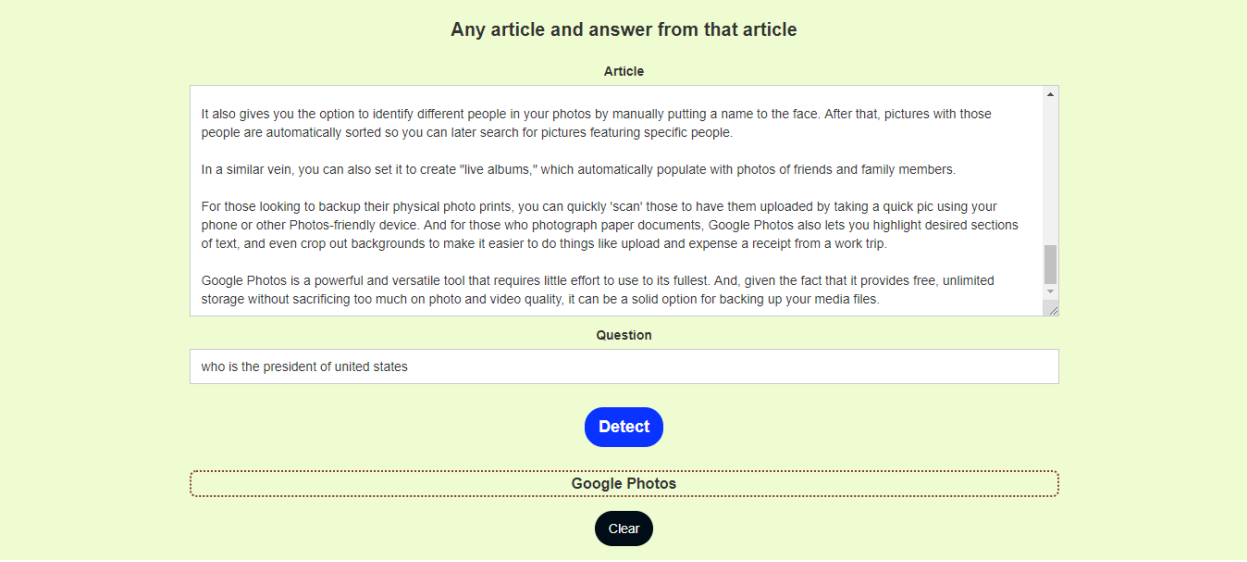
In this section artificial intelligence gives the answer based on the given article.

Section two looks like above.

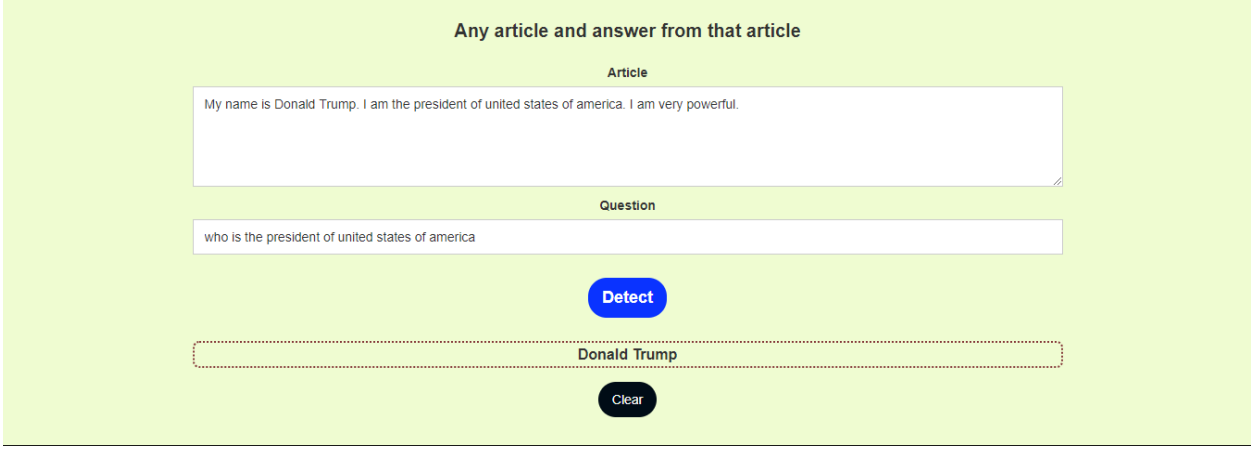
Below given some examples of how this section actually works.







The last answer is wrong. Because AI doesn't know anything else instead of the above article. That’s why it gives the wrong answer.



This time AI gives the current answer.

6.3 Search article in offline

In this section our artificial intelligence gives an article based on search query. This section works offline. Because when we trained our AI, AI saw 33758 articles. So these 33758 articles are available in the AI database. So AI can easily find the search query related article in his database.

Section three looks like the picture above.

Now below given some examples of how this section actually works.

Search article in offline

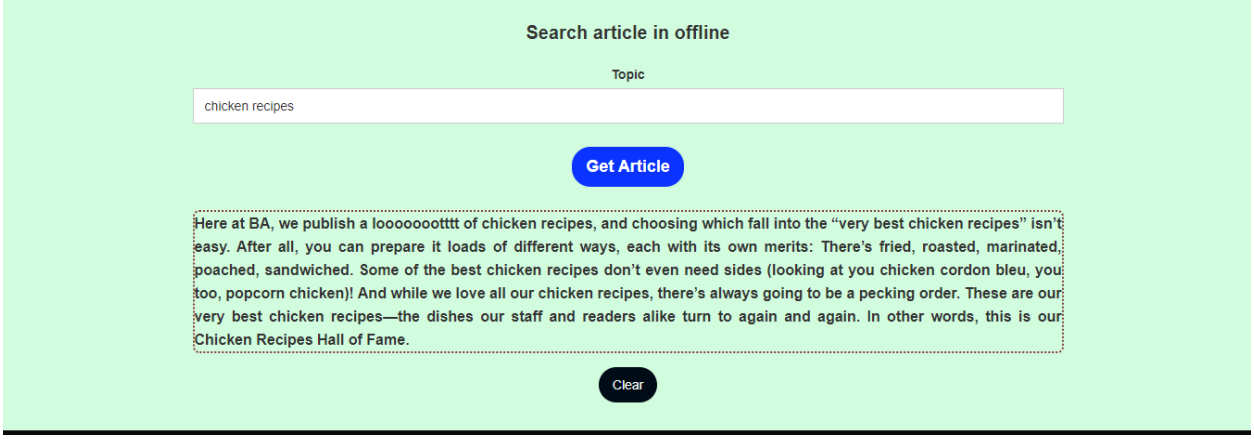
Topic

non alcoholic beer

Get Article

Non-alcoholic beer is quickly gaining traction today. But what is it and is it really safer to drink than alcohol? Today, there are a number of different brand-name and craft non-alcoholic beers to choose from, creating more options for anyone who drinks alcohol. Studies, however, have shown that this might not be the best option for some people, like alcoholics who are in recovery. Does Non-Alcoholic Beer Contain Alcohol? Non-alcoholic beer, also sometimes known as near beer is a misleading term. According to labeling regulations, non-alcoholic beer is not required to be alcohol-free and can contain some alcohol. Standard beer varies considerably in alcohol content. The average real beer contains around 5% alcohol per beer. Low-alcohol beer can contain between 0.5% and 1.2% alcohol, and non-alcoholic beer can actually contain up to 0.5 percent alcohol. There are very few, if any, non-alcoholic beers that are actually free of alcohol. In fact, some studies have shown that non-alcoholic beers may actually have more alcohol than what's listed on the label. Even if it does claim to be .5% alcohol, there may actually be quite a bit more alcohol in the non-alcoholic beer. Most will contain a small amount, which means they're not a good option for those who need to abstain from alcohol completely. Is Non-Alcoholic Beer Safe for Pregnancy? Studies have shown that alcohol can have serious impacts on the growing baby. The exact amount of alcohol that is dangerous to a fetus is unknown. What is known is that either frequent drinking of alcohol, or a single binge, can damage a growing baby. The truth is that there is no known amount of alcohol that is safe or not. A "safe" threshold has never been determined. Most women will, therefore, abstain from alcohol completely when they find out they're pregnant. However, with non-alcoholic beer, a lot of women may wonder if this is a safer option that enables them to enjoy a beer occasionally while they're pregnant without the worry that a typical beer might cause. Since non-alcoholic beers can contain some alcohol, and sometimes more than is stated on the label, it's best for anyone who is pregnant to avoid drinking non-alcoholic beer as well as regular beer. There is no known safe amount of alcohol, so those who want to stick with the rules will want to make sure they don't drink anything that could contain alcohol. A pregnant woman who is worried about not being able to drink should let their obstetrician know as soon as possible. Can I drink NA beer after I give birth? As much as you may really want a beer now that your pregnancy is over, the answer is still "probably not." After the baby is born, it's still not a good idea to drink any beer if the woman is breastfeeding. The amount of alcohol that is found in the bloodstream after drinking is the same amount that will be found in the breast milk. Long-term effects of this are still unknown, but even non-alcoholic beers can contain some alcohol and thus should be avoided while the mother is breastfeeding. Are there Risks of Drinking Non-Alcoholic Beer? There are risks associated with drinking alcohol, but are they lowered with a reduced alcohol intake? In some cases, they can be. Some studies have shown that non-alcoholic beer can help reduce the amount of time it takes for someone to fall asleep or help with anxiety. Non-alcoholic beer, nevertheless, can still contribute to liver damage. It's still not a safe option for those who are worried about liver-related medical conditions or who already suffering from medical issues with their liver. It is also dangerous to those suffering from pancreatitis. Since most alcohol is processed through the liver, even the small amount of alcohol in non-alcoholic beers can cause further damage for those who are already suffering from issues with their liver. This includes cirrhosis of the liver and a condition known as a fatty liver. Those who already have either of these conditions, other liver conditions, or who are at risk of developing these conditions will want to refrain from all alcohol, which includes non-alcoholic beers. Is Drinking Non-Alcoholic Beer Safe for Alcoholics? Once in a rare while, someone will tout non-alcoholic beer as an excellent alternative for recovering alcoholics, but this really is not the case. The claim is that the person can still enjoy a beer now and again, without worrying about actually drinking a lot of alcohol. They won't be able to get drunk, will not be at risk of driving drunk, and will significantly reduce the amount of alcohol they drink, which could help prevent medical issues they might be at risk for. The problem with this claim is that even non-alcoholic beer can trigger cravings, resulting in a relapse for many alcoholics. Triggers can occur because of the smell of the beer, the act of opening a bottle or a can, the taste of the beer, or just the idea that they're able to have a beer. These feelings remind them of when they used to drink and, without realizing it, they can easily switch back to drinking a standard beer and relapsing into their alcoholism. People in recovery Sobriety is difficult and many people have already lived for years trying to bend as many rules as possible so they can continue to drink or use drugs. If they have a non-alcoholic beer, it's just one more rule they've bent to be able to continue abusing alcohol. Though they might not be at risk for driving under the influence, they're not going to get drunk from the non-alcoholic beer, and they are significantly decreasing the amount of alcohol they're drinking, the fact is, they're still drinking alcohol. This can absolutely trigger a relapse into alcoholism or drug addiction. Another issue is that the amount of alcohol in the non-alcoholic beer might actually be more than they expect. This means they might end up drinking more than they thought they would. Even one or two drinks at night, while it won't cause them to be drunk, can be trigger cravings and a subsequent relapse. Those who want to make sure they recover from their addiction will want to make sure they avoid any alcohol at all. This means they'll want to stay away from non-alcoholic beers. Though non-alcoholic beers are quickly becoming more popular, they're not the best option for everyone. In fact, those who are pregnant, breastfeeding, dealing with liver ailments, or who are recovering from a substance abuse addiction should avoid non-alcoholic beers just like they avoid standard beers. These beers, though they're called non-alcoholic, do contain some alcohol, which means they're not an option for anyone who wants or needs to completely stop drinking alcoholic beverages. References: https://www.drinkaware.co.uk/alcohol-facts/alcoholic-drinks-units/alcoholic-and-non-alcoholic-beers/ https://www.thebump.com/a/drinking-alcohol-while-pregnant https://beerandhealth.eu/beer-and-health/health-aspects-of-non-alcoholic-beer/ https://newlifehouse.com/non-alcoholic-beer-considered-relapse/

Clear



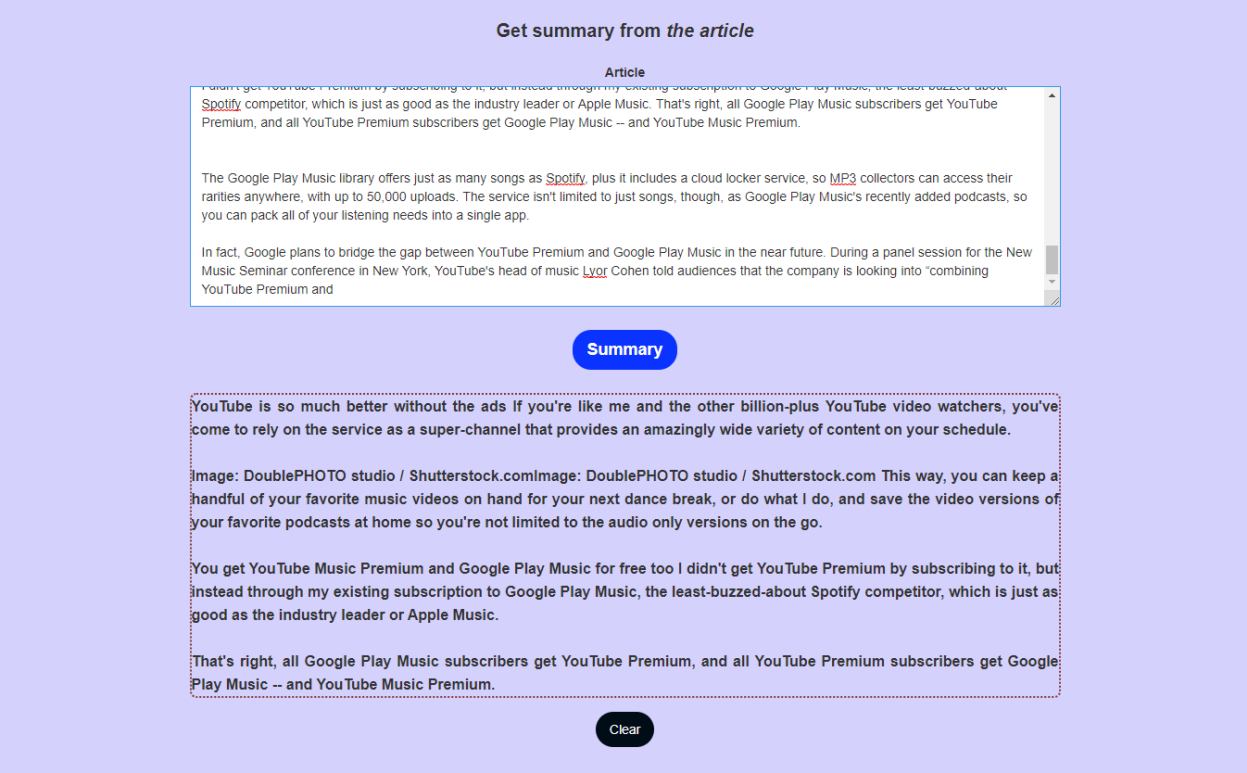
Some of the articles are not good. Because it’s totally offline.

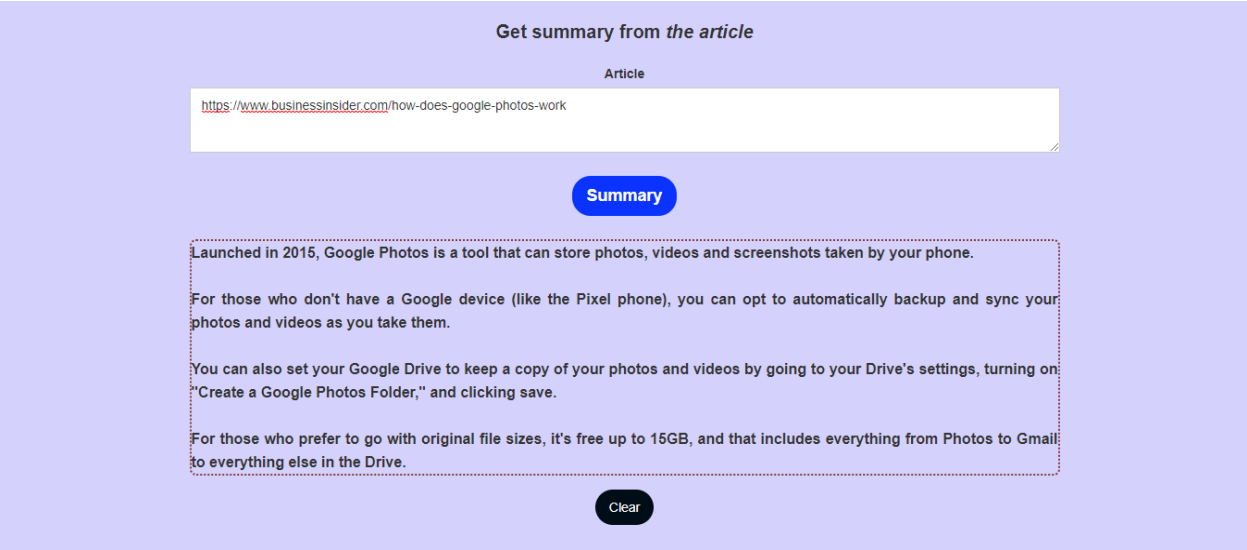
6.4 Get summary from the article

In this section AI gives the summary from the given article.

Section four looks at the picture above.

Below given some examples of how this section works.





If we just put the url of the article. It is able to give the summary from that article.

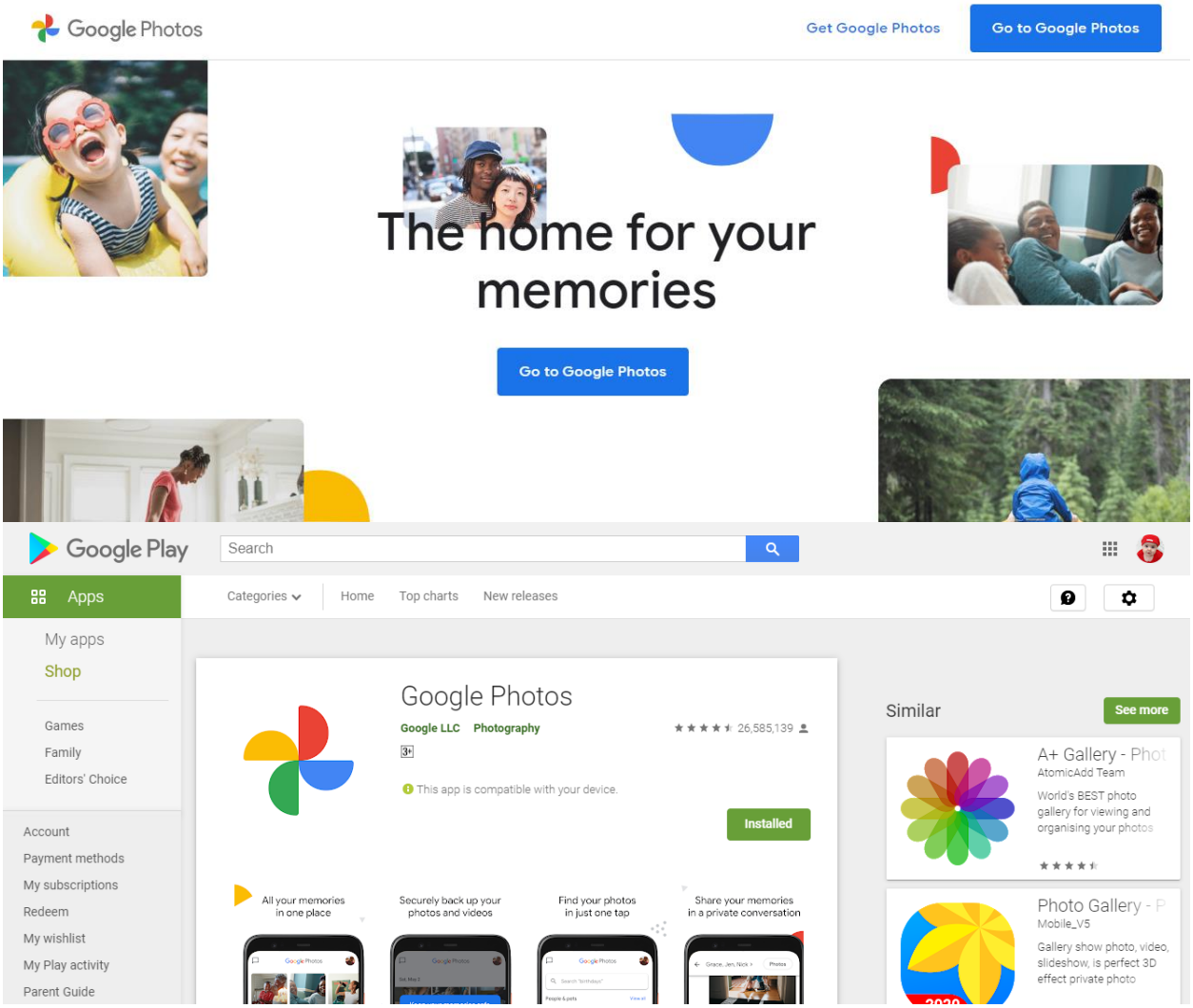
6.5 Search online and get related article

This is section five. In this section users give a search query. Then our AI searches online and finds the best article for the query.

Our section five looks at the above picture.

Now below give an example how this section works.

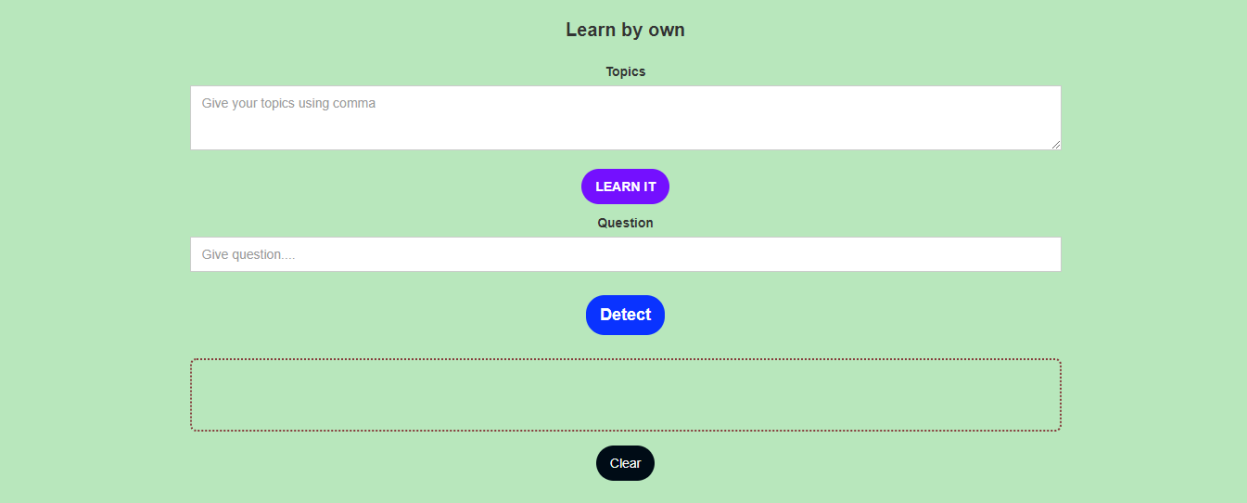
Here search query is “what is google photos”. If we search this query in google. We found the results below.



In the above two pictures is google first and second result. This result is not relevant to our question. Our question was “what is google photos”. But our AI gave a relevant article.

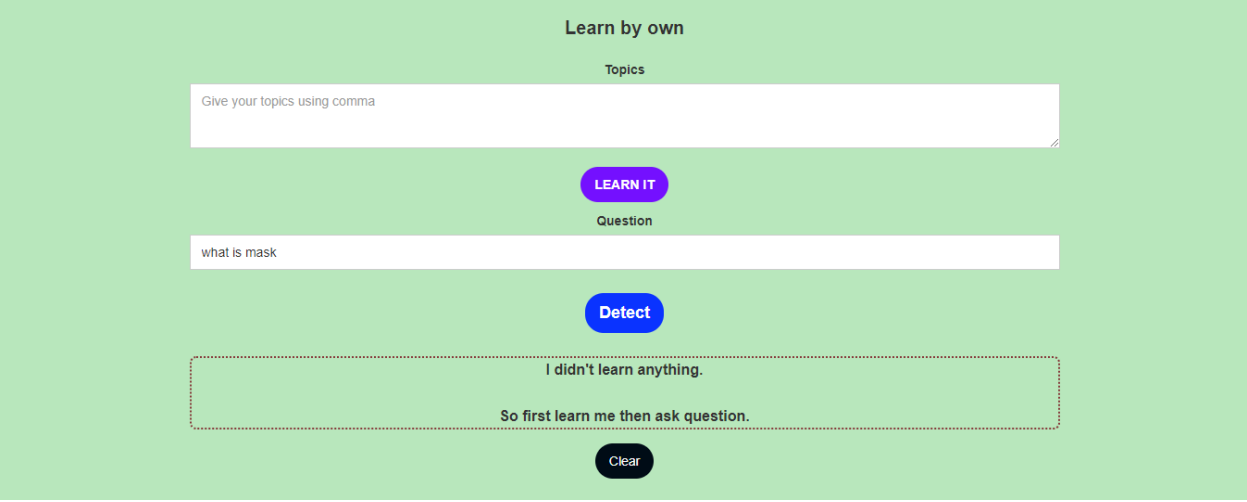
6.6 AI learn by its own

This is the last and six sections in our thesis. In this section our AI learns by own and then is able to answer the question from what AI learns.

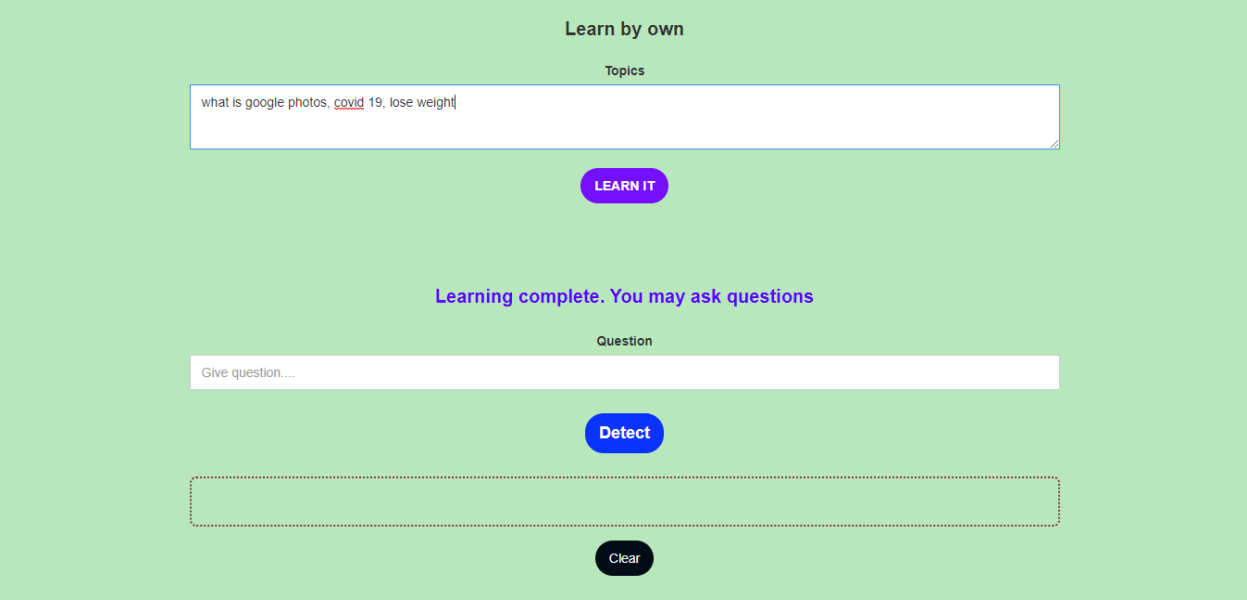


This section looks like the picture above.

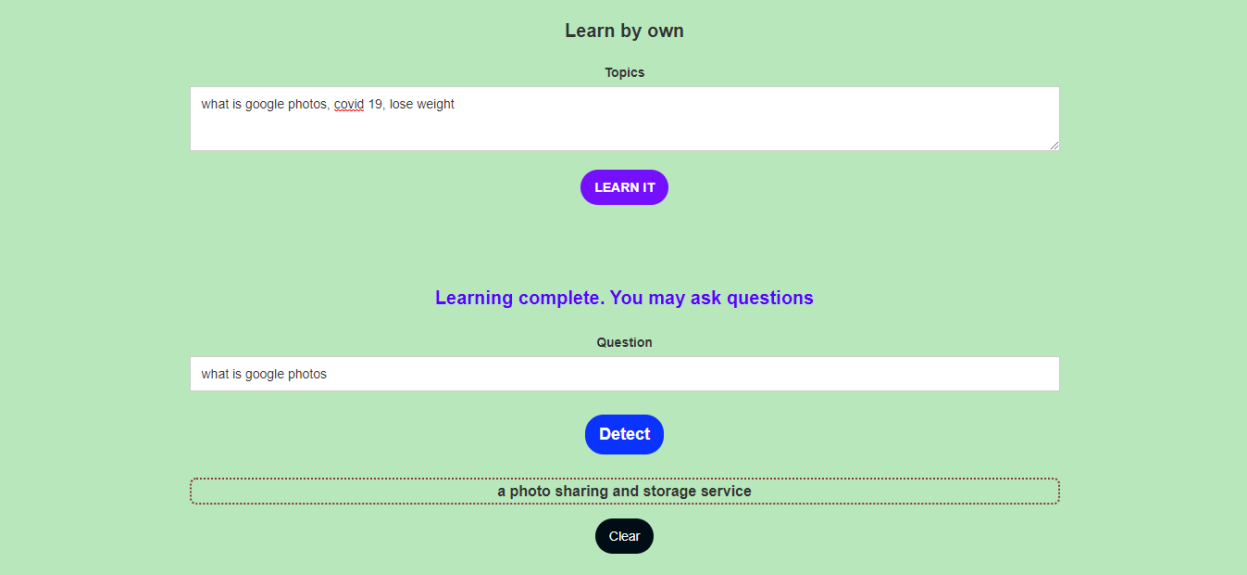
First we have to learn our AI. Otherwise AI can’t give the answer.

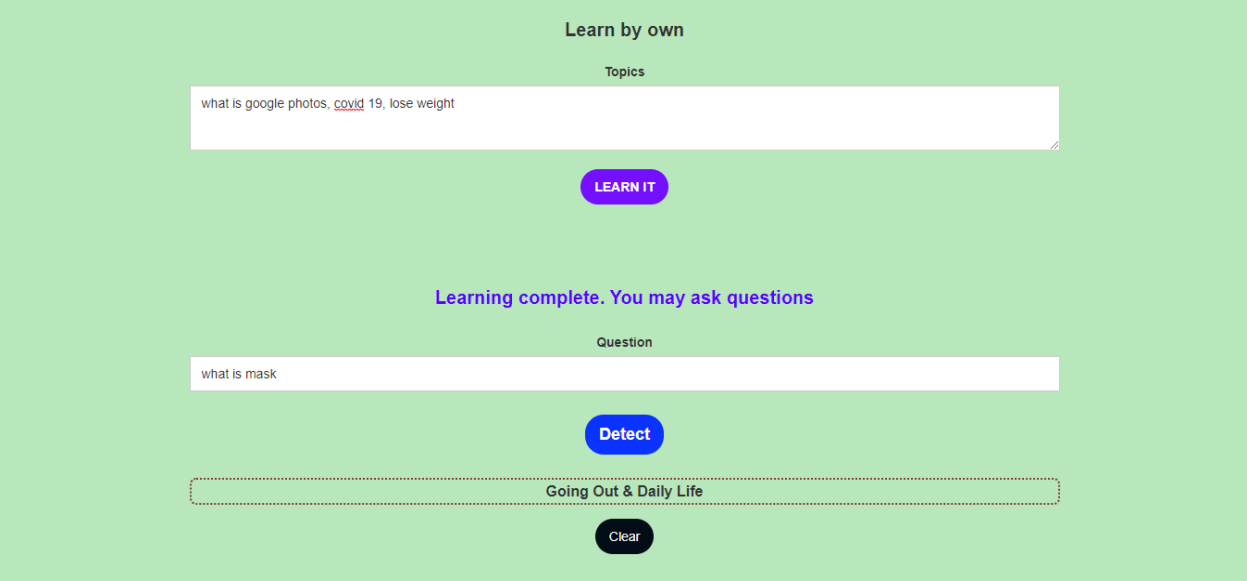


We have to give some topics in the topic box. Then AI takes some time to learn.



Now we can ask questions.





AI can only answer the question from a given topic. It doesn't know anything else.

Chapter Seven

Conclusions

Conclusions

Artificial intelligence is being used in almost every sector nowadays. Artificial intelligence has made people's lives much easier. Most of the people use artificial intelligence in their daily life. We hope our AI will help people in their daily life. We make our artificial intelligence free for everyone. We hope that our research will be useful to people and that people will take advantage of it. And in the future, we will create more artificial intelligence that will make people's daily lives easier.