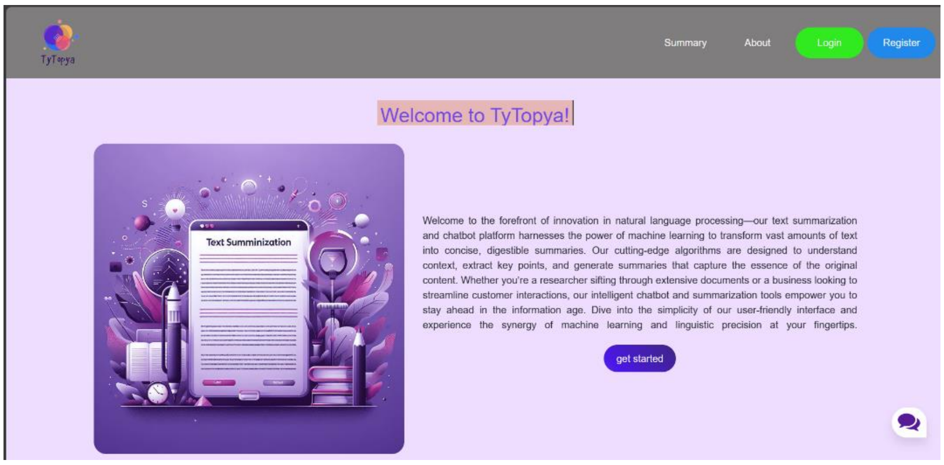
NLP Project

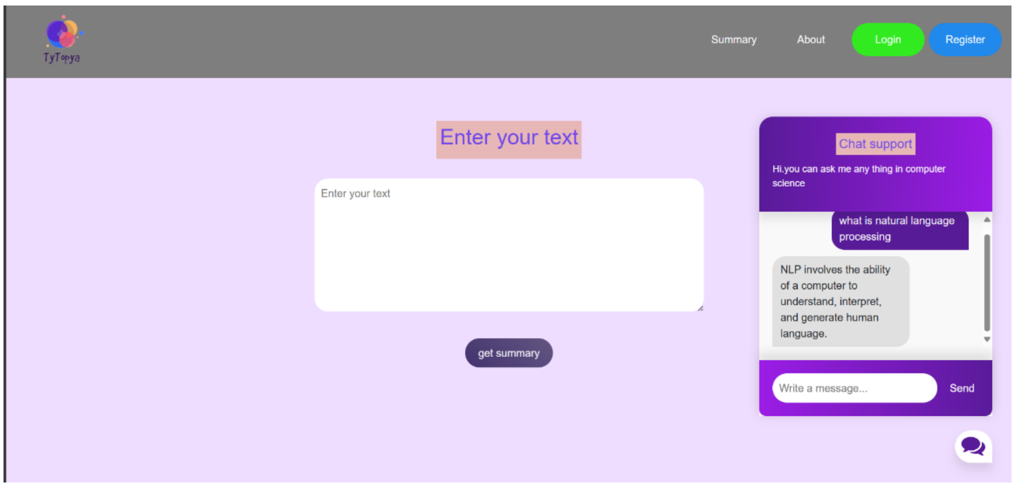
Text Summarization & ChatBot

Check_GitHub



Welcome to TyTopya!

Welcome to the forefront of innovation in natural language processing—our text summarization and chatbot platform harnesses the power of machine learning to transform vast amounts of text into concise, digestible summaries. Our cutting-edge algorithms are designed to understand context, extract key points, and generate summaries that capture the essence of the original content. Whether you're a researcher sifting through extensive documents or a business looking to streamline customer interactions, our intelligent chatbot and summarization tools empower you to stay ahead in the information age. Dive into the simplicity of our user-friendly interface and experience the synergy of machine learning and linguistic precision at your fingertips.



Hi,you can ask me any thing in computer science

what is natural language processing

NLP involves the ability of a computer to understand, interpret, and generate human language.

```
print("hello world")
```

Natural language processing: natural language processing (nlp) is an interdisciplinary field. it combines computer science, artificial intelligence, and linguistics. goal of NLP is to develop computer systems capable of "understanding" contents of documents, extracting information, and categorizing and organizing text and speech data

Deep learning: is a subset of machine learning that is inspired by the structure and function of the human brain, utilizing neural networks composed of interconnected nodes or artificial neurons organized in layers.

Transformer: transformers are a type of neural network architecture. they use an attention mechanism to selectively focus on the most relevant parts of input when generating output. transformer architectures have been widely adopted in natural language processing (NLP) some examples of transformer-based models include BERT, GPT, and t5

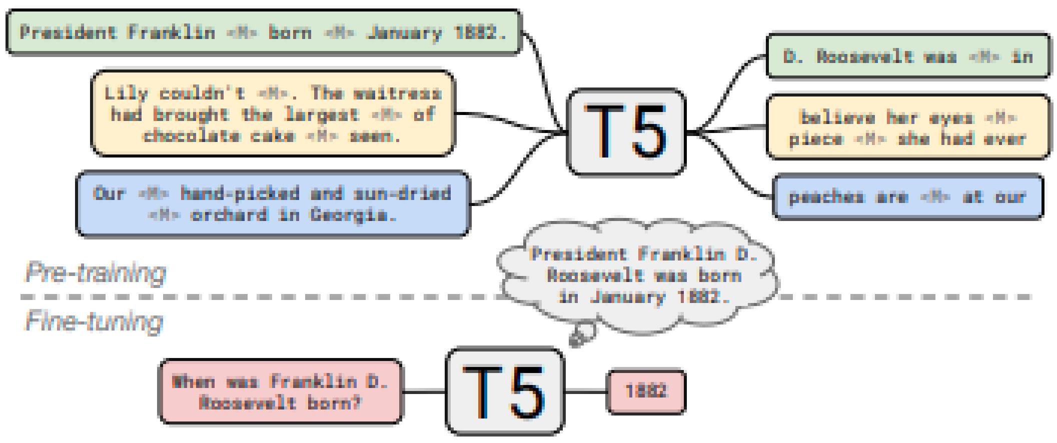
Text summarization problem: unsupervised learning & text to text transform.

ChatBot problem: deep learning & multi classification

1- Abstractive text summarization:

- Trying to understand the content of the text and then providing a summary based on that, which may or my not have the same sentences as present in the original text . abstractive summarization tries to create its own sentences and is a step towards more human-like summaries.
- Depend on: T5-large transformer model on Hugging Face
- T5-large transformer: is a pre-trained language model developed by Google researchers, introduced in the paper "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". It is a variant of the original T5 model, with improvements such as the use of the GEGLU activation function in the feed-

forward hidden layer instead of ReLU, and the disabling of dropout during pre-trainin.


President Franklin <M> born <M> January 1882.
Lily couldn't <M>. The waitress had brought the largest <M> of chocolate cake <M> seen.
Our <M> hand-picked and sun-dried <M> orchard in Georgia.
T5
D. Roosevelt was <M> in
believe her eyes <M> piece <M> she had ever
peaches are <M> at our
Pre-training
President Franklin D. Roosevelt was born in January 1882.
Fine-tuning
When was Franklin D. Roosevelt born?
T5
1882

Key Features

- Architecture: T5 is a Transformer-based Encoder-Decoder model, designed for a wide range of natural language processing tasks.
- Pre-training: The model was pre-trained on a large corpus of text data, using a text-to-text denoising generative objective.
- Improvements: The T5 v1.1 model has several improvements over the original T5, including the use of GEGLU activation and disabling dropout during pre-training.
- Language Support: The T5-large model is available in multiple languages, including Japanese and English.

- Model Size: The large version of the model has approximately 770 million parameters.
- Compute Infrastructure: The model was trained on Google Cloud TPU v3-32 infrastructure.

GEGLU: short for Generalized Gaussian Linear Units, is an advanced activation function used in deep learning models, particularly in neural networks dealing with sequential data like natural language processing and speech recognition. It is a variant of the GLU (Gated Linear Units) activation function and is designed to enhance the performance of deep learning models by capturing complex patterns in the input data more effectively.

Steps:

- Import torch ,T5Tokenizer and T5ForConditionalGeneration. Initialize t5-large and the tokenizer.
- Determine the device CPU or GPU for tokenize the text.
- Prepare and clean the text.
- Tokenize the text from string to tensor (encode)
- Get summary tensors using the model (t5-large)

- Get the summary by (decode) the tokens.

2- Extractive text summarization:

- Picking out sentences from the text that can best represent its summary. It’s more about learning to understand the importance of each sentence and their relations with each other rather than trying to understand the content of the text.
- based on Term Frequency (TF) .
- Term Frequency (TF): is a simple approach to extractive text summarization that selects the most important sentences based on the frequency of words in the document.

Steps:

- Import spacy, stop words and punctuation.
- Prepare Doc.
- Get words tokens.
- Calculate word frequences except punctuation , stop words and new line.
- Calculate max frequency of words to normalize the value.
- Score of word = Reassign each frequency / max frequency.

- Get sentences tokens.
- Score of each sentence = sum of score of each word in the sentence.
- Calculate the ratio of the summary.
- Get the most relevant sentence depending on its score and the ratio of the text using heapq model.

3- Chatbot

Intro:

A soft ware application that aims to mimic human conversation through text or voice interactions, it depends on two main tools: knowledge & understanding. Its success depends on passing Turing test. A method for determining whether a computer can think like a human being or not . one human functions as the questioner .the second human and the computer function as respondents. The questioner interrogates the respondents within a specific subject area. After a preset length of time or number of questions the questioner is then asked to decide which respondent was human and which was a computer. Steps: understanding the query – getting the answer – formatting the answer. Tools:

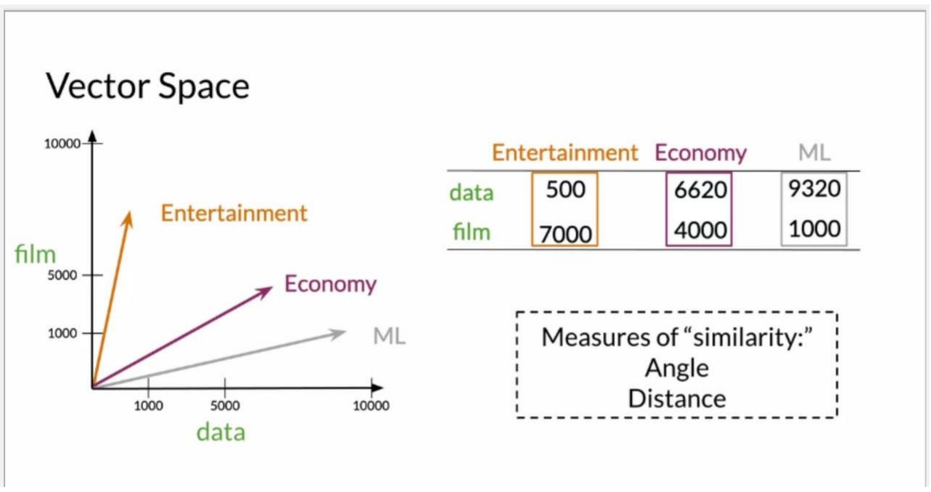
Stemming – stop-words – word embedding – auto correction – text similarity – text generation.

Some terminology :

Stemming: return each word its origin.

Stop words: all words which doesn’t change the meaning . majority of them been removed to understand the query . list of SW flexible , depends on language topic, community.

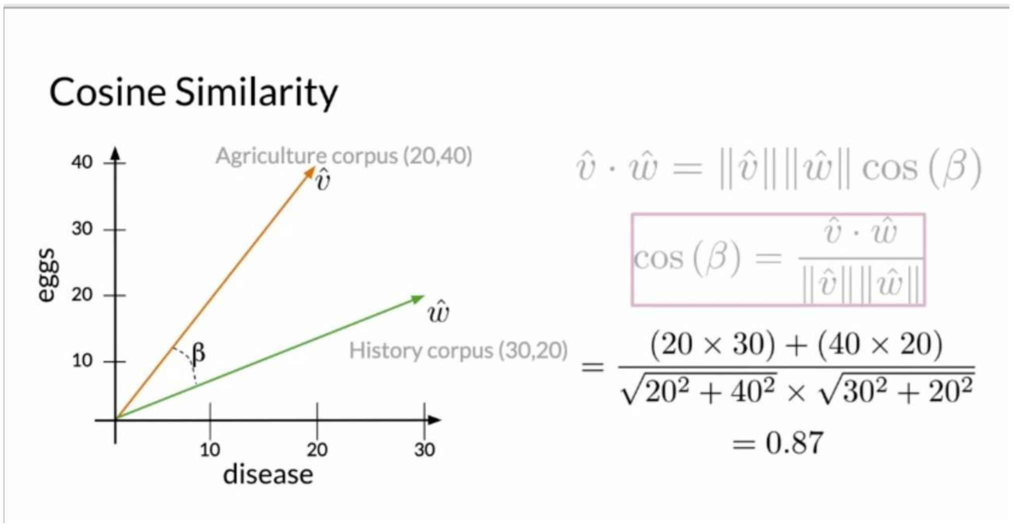
Word embedding: our idea is how much each word is close or far in meaning .it is similar to cosine similarity.



Vector Space

	Entertainment	Economy	ML
data	500	6620	9320
film	7000	4000	1000

Measures of “similarity:”
Angle
Distance



Cosine Similarity

$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

$$\boxed{\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\| \|\hat{w}\|}}$$

$$= \frac{(20 \times 30) + (40 \times 20)}{\sqrt{20^2 + 40^2} \times \sqrt{30^2 + 20^2}}$$

$$= 0.87$$

Steps to code:

1- prepare the data, the data is JSON file contains object contains a key intents which contains list of data objects Each object contains tag string, patterns list, responses list, and context_set empty string. Our data talk about ai and computer science. Object in JS like dictionary in python.

2- train file:

- Import the require libraries : nltk for data processing , Json , pickle for save results of data processing , TensorFlow ,NumPy because TensorFlow doesn't work with normal list, random to shuffle the data to don't make over fitting, download punkt for tokenizer and wordnet for lemmatization and take instance and read the Json file
- get words in all patterns after tokenization for reference
- get unique tags
- get documents which contains list of tuples each tuple contains the tokenize pattern and its tag.

- Initialize two lists training=[] and outEmpty = [0]*len(tags)
- Get each bag of each pattern after lemmatization if the word in the pattern append 1 else append 0
- Calculate the outputRow of each bag by adding reassign the of empty list by the index of the classes and get the value (0) will be 1
- Training set will be the sum of each bag + its outputRow.
- Transform it to NumPy array
- Initialize the model sequential
- Add layers of the neural network
- Initialize the optimizer SGD
- Add the loss function categorical_crossentropy and add the optimizer .
- Train the model and save in chatbot_model.h5 file.

3- chat file:

- import the require libraries load files and load models.
- Clean , tokenize and lemmatize the sentence
- Get bag of word
- Get the predictions

- Get the response

References :-

[Repo of chat bot]

[Repo of text summarization]

Introduction to Extractive and Abstractive Summarization Techniques (paperspace.com)

retrieva-jp/t5-large-long · Hugging Face

serp.ai | 526: Invalid SSL certificate

(2) The power of GeGLU in feedforward layers - Yes, improved AI with Grok-1 | LinkedIn

Removing stop words with NLTK in Python - GeeksforGeeks

Natural Language Processing (NLP) [A Complete Guide] (deeplearning.ai)

32- دورة تشات بوت Chatbot - YouTube

tytopya is a text summarization project and chatbot

tools in frontend:

- html
- CSS
- JS
- bootstrap

tools in backend:

- flask

tools in nlp:

- nltk

- spacy

tools in machine learning:

- tensorflow
- torch
- keras
- t5-large transformer of hugging face

```
print("thank you")
```