

1. **User Interface**

The interface allows user to enter the query. The query is entered into a search bar in natural language which is further preprocessed.

1. **Lowercase conversion**

All the words in user’s natural language query are converted into lowercase, making it easier to convert them into tokens.

text = text.lower()

1. **Remove punctuation**

Punctuations are removed from the natural language query as they add no info or value, reducing the size of data and increasing computational proficiency. Using regex the punctuations are removed.

import re

text = re.sub(r'[^\+\-\w\s]', '', text)

1. **Word tokenization**

The natural language query is split into minimal meaningful units referred as tokens. It is easier to extract data from the tokens. Using NLTK library the natural language query is tokenized.

!pip install nltk

import nltk

from nltk.tokenize import word\_tokenize

text\_tokens = nltk.word\_tokenize(text)

1. **Remove stopwords**

Words that have less or no meaning related to keywords are stop words. They are removed so more important keywords are focused on. Using NLTK library the stopwords can be removed.

from nltk.corpus import stopwords

stop\_words = set(stopwords.words('english'))

text\_tokens\_nostop = []

for w in text\_tokens:

if w not in stop\_words:

text\_tokens\_nostop.append(w)

1. **Spelling correction**

Natural language query entered by the user might contain spelling errors, making important data meaningless. Hence using TextBlob library spelling errors are fixed.

!pip install TextBlob

from textblob import TextBlob

txtblob = TextBlob(text)

text = str(txtblob.correct())

1. **Stemming and lemmatizing**

Stemming extracts the root word for each token. For example, “cry”, “cries” and “cried” are stemmed into “cry”. This is done by using the NTLK library.

Lemmatizing extracts the root words for tokens by checking the vocabulary.

1. **Classify elements into Relation, Attribute, Clauses**