Input: The input is any block of text from the user
Output: The output will be an automatically generated True or False sentence, with true sentences coming directly from the block of text and false sentences generated using **masking** or **negation** techniques.

Approach

1. **Coreference Resolution (optional)**

Coreference resolution (CR) is the task of finding all linguistic expressions (called mentions) in a given text that refer to the same real-world entity. After finding and grouping these mentions we can resolve them by replacing, pronouns with noun phrases.
This is optional as it's computationally heavy, but recommended. The **coref** pipeline in **spaCy experimental** is used for this. Given the input text, all pronouns and relations are resolved and outputs a resolved text.

2. **Summarization**

The given block of text or resolved text is summarized using the **distilbert** transformer model from huggingface, which returns the a shorter block of text that accurate captures the information within the text. Character limits of length 30 to 250 are placed on the returned sentences. The returned shorter text is then tokenized and by sentences using the spaCy library. These sentences form the **True** questions that are returned.

3. **Question Generation**

A sentence is then chosen at random from the returned sentences.
For **False** question generation. One of the following two methods is used at random:

- **Negation**

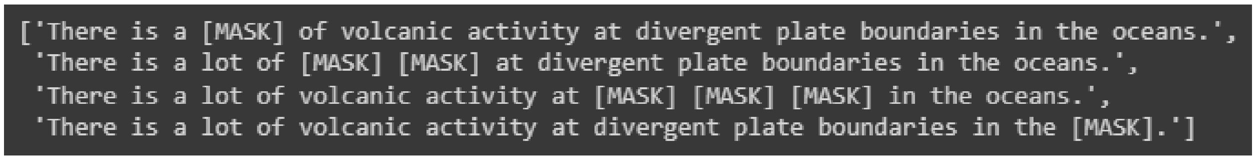
Negation involves turning a positive statement into its negative sense.
This is done using the **negate** Python module. Negate is a Python module built upon spaCy that implements rule-based, syntactic sentence negation in English.

- **Masking**

Masking involves identifying nouns phrases in the sentence and masking them. These masked sentence are then passed into a Masked Language Model which then predicts the possible tokens for the masks. The **distilbert** model finetuned on for a Masked Language Modelling task is used.

The steps involved in this are as follows:

1. Identify noun phrases in the sentence and replace each word in phrase with **[MASK].** This generates multiple sentences with different nounphrases masked.



```
['There is a [MASK] of volcanic activity at divergent plate boundaries in the oceans.',
 'There is a lot of [MASK] [MASK] at divergent plate boundaries in the oceans.',
 'There is a lot of volcanic activity at [MASK] [MASK] [MASK] in the oceans.',
 'There is a lot of volcanic activity at divergent plate boundaries in the [MASK].']
```

2. The masked sentence is fed into the MLM which predicts the words to replace the mask. This generates up to 5 possible False sentences.
3. Using cosine similarity, the 5 sentences are compared with the original sentence and the least similar is retuned. This involves generating embeddings for all sentences using a sentence transformer model from **sentence-transformers** library.

Pseudocode

```
Get input_text
if resolve_coreference:
      resolved_text =resolve coreference(input_text)
      sentence_list = summarize(resolved_text)
else:
      sentence_list = summarize(input_text)
sentence = random(sentence_list)
qtype = random([True, False])
if qtype is True:
      return sentence
else:
      method = random([masking, negation])
if method == negation:
      return negate(sentence)
else:
      masked_sentences = generate_masked_sentences(sentence)
      masked_sentence = random(masked_sentences)
      generated_falses = generate_similar_sentences(masked_sentence)
      final_sentence = get_least_similar(generated_falses)
      return final_false_sentence
```