# CROSS-LINGUAL OPEN-RETRIEVAL QUESTION ANSWERING MODEL

*Project Report Submitted in Partial Fulfilment of the Requirements for the degree of*

**Bachelor of Technology**

***in***

**Computer Science and Engineering**

*Submitted by*

Mitra Abhi Sura: (Roll No. 2010110399)
Shrey Sharma: (Roll No. 2010110600)
Vedanta Vivek Patil: (Roll No. 2010110710)

***Under the Supervision of***

Dr. Sonia Khetarpaul
Associate Professor

SHIV NADAR
INSTITUTION OF EMINENCE DEEMED TO BE UNIVERSITY
DELHI NCR

Department of Computer Science and Engineering

December, 2023

SHIV NADAR
INSTITUTION OF EMINENCE DEEMED TO BE
UNIVERSITY
DELHI NCR

NH-91, Tehsil Dadri
Gautam Buddha Nagar
Uttar Pradesh - 201314, India

Tel: +91-120-7170100
+91-120-2662002

**Date: 29 Nov 2023**

## Certificate

This is to certify that the project titled **"CROSS-LINGUAL OPEN-RETRIEVAL QUESTION ANSWERING MODEL"** is submitted by MITRA SURA (Roll No. 2010110399), SHREY SHARMA (Roll No. 2010110600) and VENDATA PATIL (Roll No. 2010110710) CSE, SNIOE in the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering.** This work was completed under the supervision of Dr. Sonia Khetarpaul. No part of this dissertation/report has been submitted elsewhere for award of any other degree.

Signature with date: 29/11/23

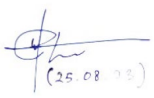
Name of Supervisor: Dr. Sonia Khetarpaul

Designation: Associate Professor

Affiliation: Shiv Nadar Institution of Eminence

# DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.



Mitra Abhi Sura  Shrey Sharma  Vedanta Vivek Patil

Date: 09/12/23

# ACKNOWLEDGEMENT

First and foremost, we would like to express our sincere gratitude to our mentor, **Dr. Sonia Khetarpaul**, for her continuous support, patience, motivation, enthusiasm, and immense knowledge. We are extremely indebted to her for all the valuable guidance and dedicated involvement in every step of the project. She taught us the methodology to carry out research and to present the research works as clearly as possible. It was from her that we learnt the importance and value of studying topics in-depth and not being satisfied with surface-level knowledge. Understanding the very sophisticated field of Information Retrieval from scratch seemed very daunting to us at first, but her vision and support throughout made the task easy. We could not have asked for a better advisor and mentor. Completing the project required more than academic support and we have our family, friends and professors to thank for listening to and, at times, tolerating us. We cannot begin to express our gratitude and appreciation for their support.

# ABSTRACT

Cross-lingual Question Answering (QA) presents a formidable challenge, requiring the ability to comprehend questions in one language and retrieve answers from documents in another. It is motivated by the fact that many languages face both information scarcity and information asymmetry.

We propose a novel approach that uses helsenki transformers to convert the query into various languages and combines BM25, a traditional information retrieval model, with BERT, a state-of-the-art language model. We leverage BM25 for efficient document retrieval and then employ BERT for contextual understanding and answer extraction. In our project, we utilized T5 for the systematic generation of questions, which were then utilized to evaluate the pretrained model.

Building upon this foundation, we investigate the concept of zero-shot cross-lingual QA and Knowledge Base (KB) comprehension. Our work contributes to bridging language barriers in information access and retrieval, making knowledge more accessible and inclusive across linguistic diversity.

# CONTENTS

**1.** Introduction ..... 9

**2.** Literature Review ..... 11

2.1 XOR QA: Cross-lingual Open-Retrieval system ..... 11

2.2 Zero Shot Cross-Lingual Transfer ..... 13

2.3 Using Knowledge base as Reading Comprehension (xKBQA) . 16

**3.** Project Outline ..... 19

**4.** Dataset and Tools ..... 21

4.1 XOR-TYDI QA ..... 21

4.2 MKQA ..... 21

4.3 TYDI QA ..... 21

4.4 BM25 ..... 22

4.5 BERT ..... 22

4.6 Universal Sentence Encoder ..... 23

4.7 T5 Question Genration Model: ..... 23

4.8 mBERT (Multilingual BERT): ..... 23

4.9 BERT Working ..... 24

4.10 Helsinki-NLP ..... 32

**5.** Methodology ..... 35

5.1 SDLC Approach (software development lifecycle) ..... 35

5.2 Working on Mandarin Language ..... 35

5.3 Approach: BERT ..... 36

5.4 Approach: BM25 + BERT ..... 36
5.5 Approach: BERT Model Caching ..... 39
5.6 QA Generation and Model Fine Tuning ..... 40
5.7 Fine Tuning on SQuAD Dataset ..... 42
5.8 Building an in-hand translator. ..... 43
5.9 Approach: BM25 + BERT + Cross Lingual ..... 44

**6.** CONCLUSION AND RESULT ..... 46

**7.** PROJECT UI ..... 51

**8.** FUTURE SCOPE ..... 53

**9.** REFERENCES ..... 54

# LIST OF FIGURES


2.1.1 Overview of XOR QA ..... 8
2.1.2 Overview of Tasks and Baselines ..... 12
2.1.3 Overview of the annotation process for XOR-TyDI QA ..... 12
2.2.1 Pretrained Multilingual Zero-Shot Translation ..... 13
2.2.2 Zero-Shot translation Pipeline ..... 14
2.3.1 Cross lingual query example ..... 16
2.3.2 Working of answer prediction using Zero-Shot Baselines ..... 17
3.1 Project Outline ..... 19
4.1 Question-Answer pairs using XOR-TYDI QA ..... 21
4.2 Question-Answer pairs using MKQA ..... 21
4.3 Question-Answer pairs using TYDI QA ..... 21
4.4 BM25 Scoring ..... 22
4.5 BERT Scoring ..... 22
4.6 Scoring methods for sparse and dense vectors ..... 22
4.7 Universal Sentence Encoder ..... 23
4.8 T5 Question Generation model ..... 23
4.9 BERT Transformer Architecture ..... 24
4.10 BERT Version ..... 25
4.11 Tokenization using BERT ..... 26
4.12 BERT Segmentation Embeddings ..... 27
4.13 Embeddings to QKV vectors ..... 27
4.14 Learned weights using QKV vectors ..... 28
4.15 Calculating Attention using QKV ..... 28
4.16 Multi-Head Attention Layers ..... 29
4.17 Final Matrix Calculation ..... 29
4.18 Answer span prediction ..... 31

4.19 Helsinki-NLP ..... 32
5.1.1 Spiral Model lifecycle ..... 35
5.2.1 Mandarin document snippet ..... 35
5.2.2 Document identification using BERT-base Chinese ..... 35
5.4.2.1 Text tokenization code snippet ..... 36
5.4.3.1 BM25 calculation code snippet ..... 37
5.4.4.1 QA-pipeline code snippet ..... 37
5.4.6.1 Document wise answer span results ..... 38
5.5.1 Project file structure ..... 39
5.5.2 Result for user manuals ..... 39
5.6.1 T5 Model Pipeline ..... 40
5.6.2 Saving data in JSON ..... 41
5.7 Model Training ..... 42
5.8.1 In hand transalator ..... 44
5.9.1 Cross lingual Model ..... 45
6.1 Precision and recall ..... 46
6.2 F1 Score ..... 47
6.3 Bar graph of precision, recall and F1 score for all 5 languages ..... 47
6.4 Sample similarity score for a query ..... 48
6.5 Bar graph to compare scores among languages ..... 48
6.6 Heatmap for Table 6.1. ..... 50
7.1 Front page of the Cross Lingual Model withal the functionalities ..... 51
7.2 A query is entered to get the result in different languages. ..... 52

# LIST OF TABLES

6.1 Similarity scores for all languages..... 49

# ABBREVIATIONS

**BERT:** Bidirectional Encoder Representations from Transformers

**T5:** Text-To-Text Transfer Transformer

**XOR TYDI QA:** Cross-lingual Open Retrieval Question Answering

**QKV vectors:** Query, Key, Value vectors

**NLP:** Natural Language Processing

**NSM:** Neural Semantic Matching

**KBQA:** Knowledge Base Question Answering

**CLS token**: Classification Token

**SDLC:** Software Development Life Cycle

**BM25:** Best Matching 25

**USE:** Universal Sentence Encoder

# INTRODUCTION

**Motivation:**

In an era of globalized communication and information exchange, the need for effective tools that transcend linguistic boundaries has become increasingly evident. As part of this landscape, our project focuses on the development of a cross-lingual open retrieval question-answering (QA) model, aiming to facilitate information access and cross-cultural communication. This section explores the motivations behind the project, emphasizing the broader context of linguistic diversity and the challenges it presents.

Our project on developing a cross-lingual open retrieval question-answering (QA) model is motivated by the imperative for a more inclusive and globally accessible information landscape. In a world marked by linguistic diversity, our model seeks to overcome language barriers and enhance the user experience by allowing individuals to interact with digital content in their native languages. This endeavor is rooted in the understanding that language is not merely a means of communication but a vessel for cultural nuances. By promoting cultural inclusivity, our model empowers users to access information reflecting diverse perspectives.

Furthermore, in addressing the challenges posed by linguistic diversity, our project contributes to facilitating multinational collaborations and supporting market expansion for businesses operating on a global scale. A key aspect of our approach involves optimizing resource utilization by leveraging knowledge across languages, providing a scalable solution to the complexities of linguistic diversity. In essence, our cross-lingual QA model aims to bridge linguistic gaps, foster cross-cultural communication, and contribute to a more connected and universally accessible digital environment.

This project aims to address these challenges by creating a cross-lingual open-retrieval question answering model that can answer questions across languages. Existing models often struggle with language-specific limitations and the inability to handle multiple languages effectively. Our approach integrates advanced natural language processing

techniques, state-of-the-art models, and cross-lingual capabilities to create a robust and versatile system.

We have made use of BERT, short for Bidirectional Encoder Representations from Transformers, is a pre-trained natural language processing model developed by Google. It's known for its effectiveness in various NLP tasks and is designed to consider the context of a word or phrase by examining both the left and right context simultaneously.

"Hugging Face BERT Base Chinese" specifically refers to a BERT model pre-trained on Chinese text data to tokenize Chinese text for adding cross lingual capabilities and language specific pre-training. We have explored a range of datasets to facilitate comparative analysis among language- based models. XOR QA is a cross-lingual open-retrieval question answering (QA) system that can answer questions across languages. XOR QA was able to handle a wide range of languages, including low-resource languages, and was able to answer questions from a variety of domains. TyDI QA is another multilingual QA dataset that contains questions and answers in multiple languages, including English, French, German, Spanish, Chinese, Japanese, and Korean.

T5, which stands for "Text-to-Text Transfer Transformer," is a versatile and pre-trained model for various NLP tasks. In this context, T5 was used to generate questions based on provided context. This approach simplifies the process of creating question-answering datasets. Parallely, we have used various evaluation metrics to assess the performance of models in information retrieval and question answering using BM25 and TF-IDF.

This project leveraged the Helsinki-NLP's Opus-MT models for translation in our cross-lingual open retrieval question-answering (QA) model. Specifically, we employed these pre-trained models to facilitate seamless translation between languages, enabling users to interact with the QA system in their preferred language. To enhance the accuracy and relevance of our QA model, we utilized T5 (Text-To-Text Transfer Transformer) to generate a curated dataset for training and verification. This dataset played a pivotal role in fine-tuning our model, ensuring its proficiency in understanding and responding to user queries across diverse languages. The combination of Helsinki-NLP's translation capabilities and T5-generated QA datasets forms a robust foundation for the cross-lingual functionality of our model.

# LITERATURE REVIEW

## 2.1) XOR QA: Cross-lingual Open-Retrieval system

The paper "XOR QA: Cross-lingual Open- Retrieval Question Answering" by Artetxe et al. (2021) introduces a new task framework called Cross-lingual Open- Retrieval Question Answering (XOR QA), which consists of three new tasks involving cross-lingual document retrieval from multilingual and English resources as shown in fig. 2.1.1

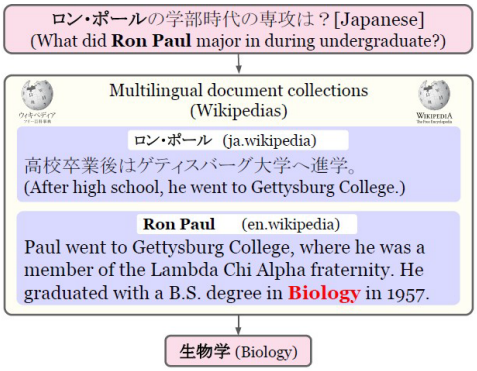



**Figure 2.1.1) Overview of XOR QA, given a question in Li and finds the answer in English or Li**

The authors evaluate a variety of approaches on XOR QA and find that the best performing approaches use a combination of machine translation and cross-lingual pretrained models.

The three tasks in XOR QA are:

XOR-Retrieve: This task is to retrieve the top 50 English Wikipedia paragraphs that answer a question in a target language.

XOR-English Span: This task is to output a short answer in English to a question in target language.

XOR-Full Answer: This task is to output a full answer in English to a question in a target language, which may include multiple paragraphs and sentences.

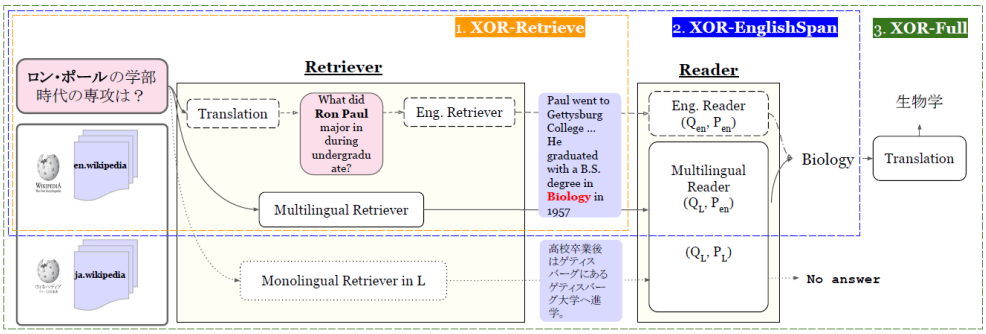



**Figure 2.1.2)** **Overview of the tasks and baselines** **[4]**

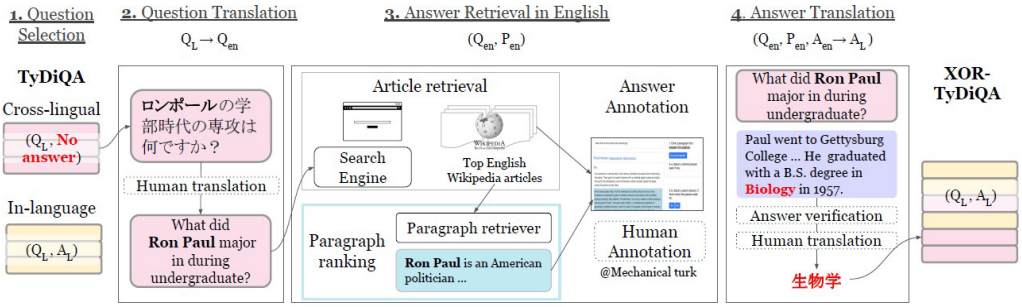



**Figure 2.1.3)** **Overview of the annotation process for XOR-TyDI QA** **[4]**

1. **Question Selection:** Randomly selecting 5,000 unanswerable questions from TYDI QA, the authors split them into training and development sets across seven languages.

2. **Question Translation:** Using Gengo translation service, questions are professionally translated into English, with a focus on accurately rendering named entities.

**3a. Answer Retrieval in English:** Employing Amazon Mechanical Turk, the study retrieves answers for translated English questions from Wikipedia, introducing a collaborative model-in-the-loop framework to balance retrieval and annotation errors.

**3b. Quality Control for QA Annotation:** High-approval-rate MTurkers from English-speaking countries are recruited, and a qualification batch is used to select high-quality annotators out of more than 200 workers.

4. **Answer Verification and Translation:** Undergraduate students verify answers, with only 8% marked as incorrect, corrected later by high-quality crowd workers. Gengo is again used for translating answers back to the original languages,

focusing on consistency with TYDI QA.

5. **Data Mixing:** To represent actual question distributions, annotated cross-lingual data is mixed with same-language data from TYDI QA, reflecting a combined source for diverse native speaker contexts.

## 2.2) Zero Shot Cross-Lingual Transfer

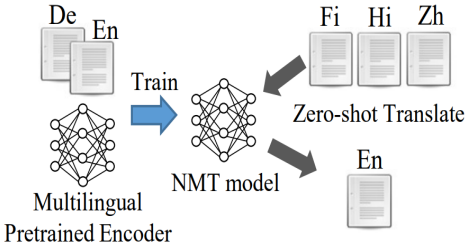



**Fig 2.2.1** Pretrained Multilingual Zero-Shot Translation [8]

Zero-shot cross-lingual transfer is the ability of a model to perform well on a task in a new language without any additional training on that language. This is a highly desirable property for cross-lingual NLP models, as it allows them to be used for tasks in a wide range of languages without the need to develop and train separate models for each language as shown in fig. 2.2.1.

One way to achieve zero-shot cross-lingual transfer is to use multilingual representations. Multilingual representations are trained on a corpus of text in multiple languages, and they learn to represent the meaning of words and phrases in a way that is generalizable across languages. This allows multilingual representations to be used for cross-lingual tasks without any additional training.

However, multilingual representations can be expensive and time-consuming to train, and they may not be suitable for all languages, especially those with limited resources.

**Language-specific Position Embeddings:**

- **Modification:** Learns separate position embeddings for L2 in Step 2, addressing word order variations in different languages.

- **Special Case:** Excludes position and segment embeddings for the [CLS] symbol, as it doesn't contribute additional capacity.

**Noised Fine-tuning:**

- **Adjustment:** Adds Gaussian noises during fine-tuning in Step 3 to enhance model robustness to the discrepancy between training with L1 embeddings and testing with L2 embeddings.

**Adapters:**

- **Innovation:** Introduces residual adapters to improve deep representations of L2 while preserving alignment with L1.

- **Implementation:** Adapter modules are added strategically after specific layers during the transfer from L1 to L2 in Step 2, with trainable parameters.

These modifications aim to enhance cross-lingual model performance by addressing issues related to position embeddings, fine-tuning, and adapter modules (refer to figure 2.2.2).

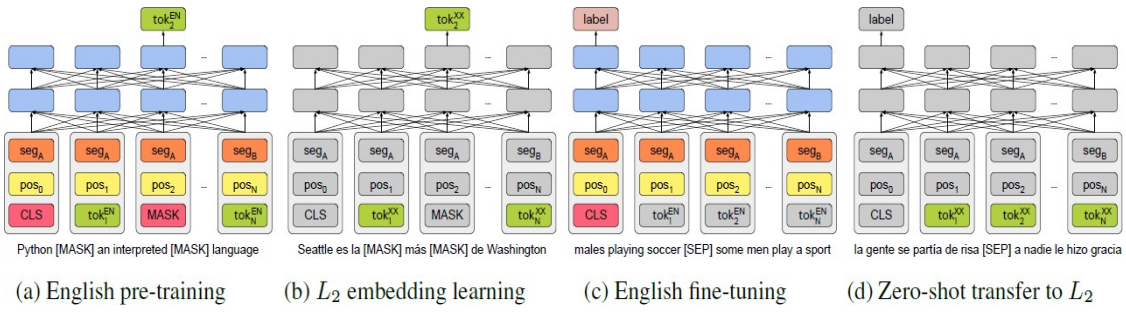



**Fig 2.2.2** Zero-Shot translation Pipeline [8]

**The steps are as follows:**

a) **Pre-train a monolingual transformer** model in English akin to BERT. This involves training a transformer-based masked language model on a large corpus of English text.

b) **Freeze the transformer** body and learn new token embeddings from scratch for a second language using the same training objective over its monolingual corpus. This involves freezing the parameters of the transformer model and training a new embedding matrix for the second language using the same masked language modelling objective as in step 1.

c) **Fine-tune the model** on English while keeping the embeddings frozen. This involves fine- tuning the model on a supervised task in English, such as natural language inference, while keeping the embedding matrix for the second language frozen.

d) **Zero-shot transfers** it to the new language by swapping the token embeddings. This involves swapping the embedding matrix for English with the embedding matrix for the second language.

This paper shows that it is possible to achieve good zero-shot cross-lingual performance by simply transferring a monolingual model to a new language at the lexical level, without the need for joint training or a shared vocabulary. This finding has important implications for the development of cross-lingual NLP models, especially for languages with limited resources, even without the use of deep multilingual abstractions.

## 2.3) Using Knowledge base as Reading Comprehension (xKBQA)

This paper aims to answer questions in languages different from that of the provided knowledge base (KB). One of the major challenges facing xKBQA is the high cost of data annotation, leading to limited resources available for further exploration.

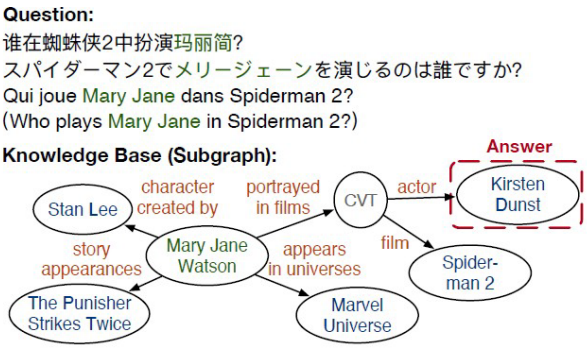



**Figure 2.3.1** Cross-lingual query example [25]

Another challenge is mapping KB schemas and natural language expressions in the questions under cross- lingual settings.

The system works as follows (refer to fig. 2.3.1):

1. Given a question in a language *L*1 and a knowledge base (KB) in a language *L*2, the system first extracts a subgraph from the KB according to entity linking results.

2. The system then converts the subgraph into a question-specific passage in natural language using a KB-to-text generation model.

3. The question and passage are then fed into a multilingual pre-trained language model (MPLM) to predict the answer to the question.

The xKBQA system based on reading comprehension has a number of advantages over traditional xKBQA systems. First, it is able to answer questions in a language different from that of the KB, without the need to translate the KB into the language of the question. Second, it can learn from a large corpus of text

in multiple languages, which allows it to better understand the context of the question and to generate a more accurate answer as explained in fig. 2.3.2.

**Rationale:**

- **Supervised Baselines:** Utilize established methods of translating non-English data into English for evaluation, covering both information extraction and semantic parsing styles.

- **Zero-shot Baselines:** Address scenarios where supervised baselines lack training data, introducing novel approaches such as Multilingual Semantic Matching and BLI for zero-shot evaluation.

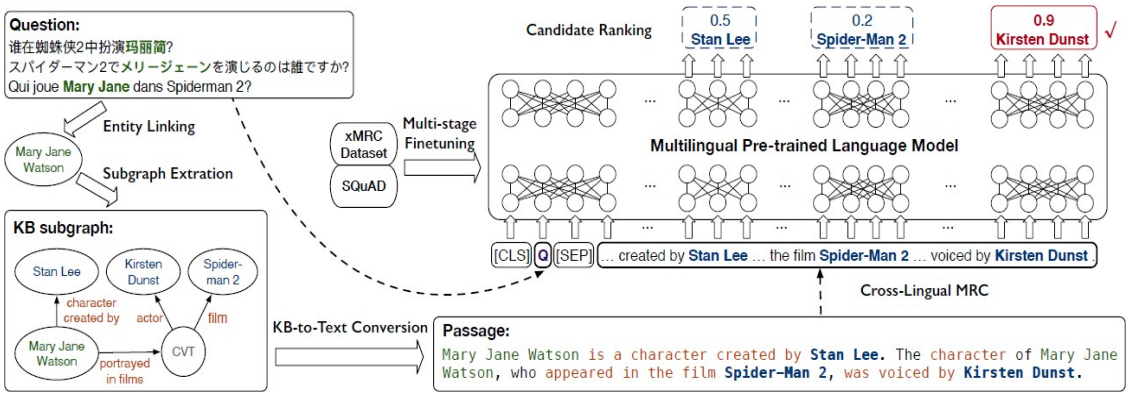



**Figure 2.3.2** Working of answer prediction using Zero-Shot Baselines [25]

**Supervised Baselines:**

- **English-as-pivot Method:**
    - **Approach:** Translating non-English language data into English using machine translation tools and utilizing monolingual models.
    - **Models Evaluated:** EmbedKGQA, GraftNet, NSM, and QGG for information extraction style.
    - **Closed-book QA Baseline:** Evaluates models like mT5, with questions directly fed into the model for answer generation without external knowledge like Knowledge Bases (KBs).

**Zero-shot Baselines:**

- **Multilingual Semantic Matching:**
    - **Approach:** Measures similarity between questions and inferential chains using an MPLM fine-tuned on LCQuAD, an English KBQA dataset.
- **Bilingual Lexicon Induction (BLI):**
    - **Approach:** Augments data through word-to-word translation using BLI, based on the Multilingual Semantic Matching baseline.

This provides a comprehensive overview of baseline methods, encompassing both supervised and zero-shot approaches, to assess the performance of cross-lingual QA models.

# PROJECT OUTLINE

Our methodology is designed to address the key components of a cross-lingual open- retrieval question answering model:

**3.1 Data Collection**: Gathering a diverse dataset of questions, answers, and documents from various languages, including but not limited to XOR-TYDI QA, MKQA, and TYDI QA.

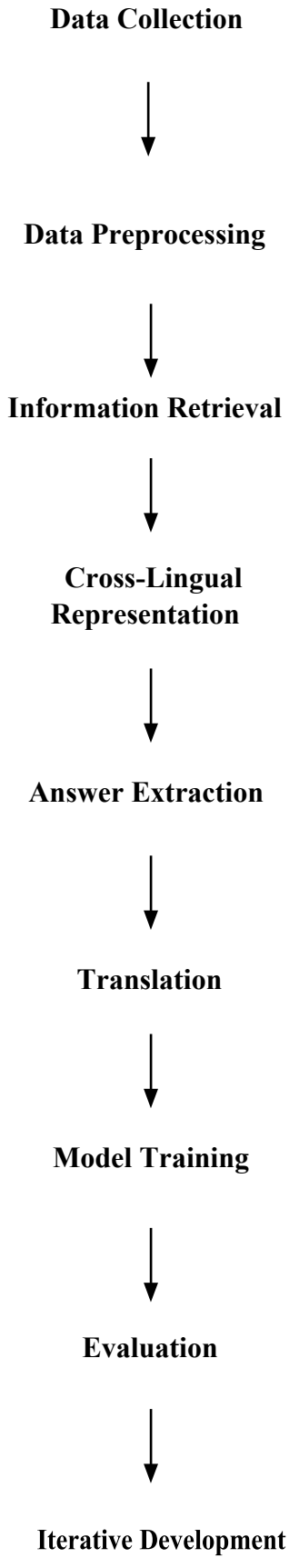
**3.2 Data Preprocessing**: Cleaning and preparing the collected data by eliminating unwanted characters, normalizing text, case folding, removing stop words, and lemmatization.

**3.3 Information Retrieval**: Developing retrieval mechanisms to find relevant documents for a given question.

**3.4 Cross-Lingual Representation**: Using multilingual models like mBERT and techniques to represent text in a language-agnostic way.

**3.5 Answer Extraction**: Extracting answers from retrieved documents using span prediction techniques.

**3.6 Translation**: Incorporating translation mechanisms for handling cross-lingual queries.



**3.7 Model Training**: Fine-tuning a pretrained model (e.g., mBERT) on specific cross-lingual question-answering datasets.

**3.8 Evaluation**: Employing various evaluation metrics such as accuracy, precision, recall, F1 match, and MRR (Mean Reciprocal Rank) to assess model performance.

**3.9 Iterative Development**: Continuously refining the model based on evaluation results and feedback.

# DATASETS

**4.1 XOR-TYDI QA**: This dataset serves as a valuable resource for cross-lingual question answering, particularly in low-resource languages. It focuses on providingquestions and answers in a variety of languages, enabling the development of systems that can handle linguistic diversity effectively. Refer to fig. 4.1 to view the QA format.


{"id": "-2745676532874802308", "question": "উইকিলিকস কত সালে সর্বপ্রথম ইন্টারনেটে প্রথম তথ্য প্রদর্শন করে ?", "answers": ["2006"], "lang": "b
{"id": "-1876566410524927695", "question": "দ্বিতীয় বিশ্বযুদ্ধে কোন দেশ পরাজিত হয় ?", "answers": ["Germany"], "lang": "bn", "split":


Figure 4.1 Question-answer pairs using XOR-TYDI QA [24]

**4.2 MKQA (Multilingual Knowledge Questions and Answers)**: MKQA is a multilingual question-answering dataset designed to encompass diverse language types. This dataset (as shown in fig. 4.2) is instrumental in testing the system's proficiency in handling linguistic variations and ensuring reliable answers for users worldwide.

| example_id<br>string | queries<br>dict | query<br>string | answers<br>dict |
|---|---|---|---|
| 3051930912491995402 | { "ar": ": كم من الوقت استغرق بناء البرجين التوأمين", "da": "hvor lang ti… | how long did it take the twin towers to be built | { "ar": [ { "type": 5, "entity": "", "text": "11.0 سنة", "aliases"… |
| 6277735658261425592 | { "ar": "من أين جاءت العبارة \"great scott\"", "da": "hvor stammer udråbet… | where did the phrase great scott come from | { "ar": [ { "type": 1, "entity": "", "text": null, "aliases": [] }… |

Figure 4.2 Question-answer pairs using MKQA [2]

**4.3 TYDI QA (Typologically Diverse Question Answering)**: TYDI QA is a dataset that consists of questions and answers in numerous languages. Encompasses questions and answers in numerous languages, spanning 11 language families and 26 scripts, showcasing remarkable linguistic diversity as show in fig. 4.3.

Q: ما هي اول سيمفونية لبيتهوفن ؟
? lbythwfn symfwnyp Awl y mA
*What is Beethovens first symphony?*

A: 21 Op. تصنيف الكبير دو سلم في لبيتهوفن الأولى لسيمفونية
21 Op. tSnyf Alkbyr dw slm fy lbythwfn AlA#wlY lsymfwn
*Beethoven's First Symphony in C Major Op. 21*

Figure 4.3 Question-answer pairs using XOR-TYDI QA [24]

# TOOLS

**4.4 BM25 (Best Matching 25)**: BM25 is an information retrieval algorithm that leverages a bag-of-words model to score documents according to their relevance to the query. We will use BM25 for document retrieval, allowing our system to efficiently identify potential sources of information for a given question. Refer to fig. 4.4 for more.

$$BM25 = \sum_{t \in q} \log\left[\frac{N}{df(t)}\right] \cdot \frac{(k_1 + 1) \cdot tf(t,d)}{k_1 \cdot \left[(1-b) + b \cdot \frac{dl(d)}{dl_{avg}}\right] + tf(t,d)}$$

- $k_1$, $b$ – parameters
- $dl(d)$ – length of document $d$
- $dl_{avg}$ – average document length

Figure 4.4 BM25 Scoring [10]

**4.5 BERT (Bidirectional Encoder Representations from Transformers)**: It is a pre-trained natural language processing model known for its effectiveness in various NLP tasks and is designed to consider the context of a word or phrase by examining both the left and right context simultaneously as shown in fig. 4.5.

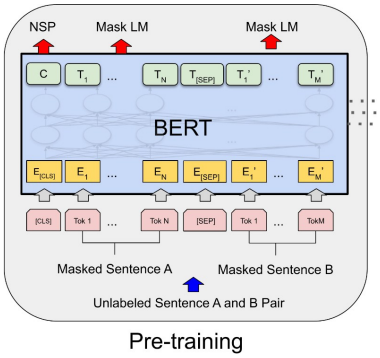



Figure 4.5 BERT Encoder [21]

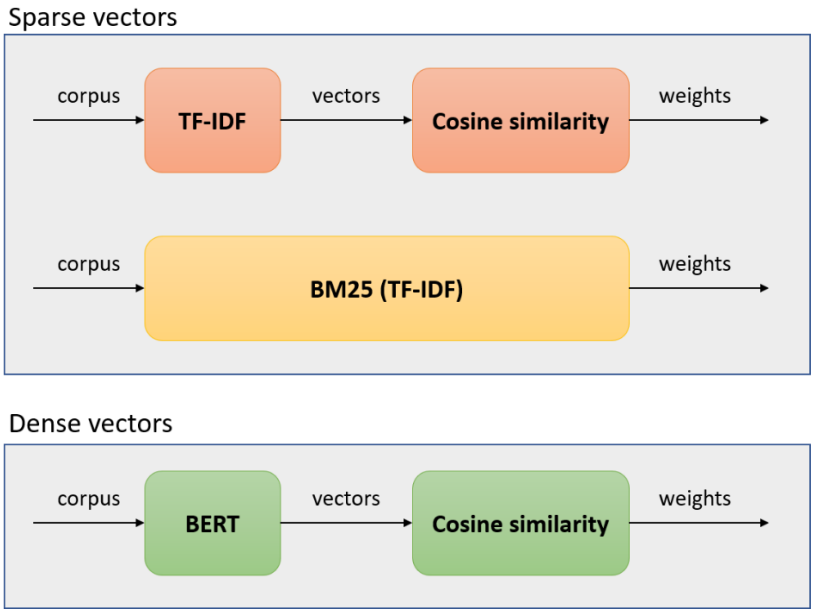



Figure 4.6 Scoring methods for Sparse and dense vectors [23]

**4.6 Universal Sentence Encoder (USE):** It is a sentence embedding model that can represent sentences in a language-agnostic way. USE is trained on a massive dataset of text and code, and it can be used to compute the similarity between sentences in different languages. Refer to fig. 4.7 for a visual representation.

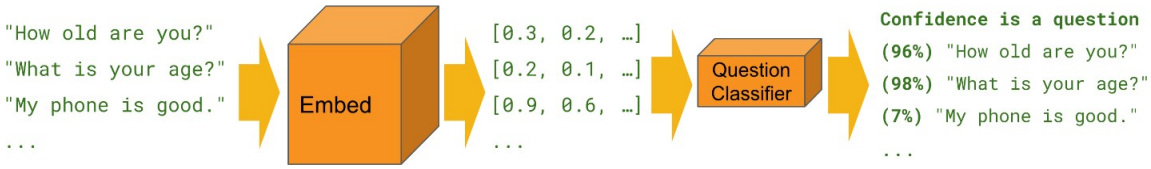



Figure 4.7 Universal Sentence Encoder [20]

**4.7 T5 Question Genration Model:** Which stands for "Text-to-Text Transfer Transformer," is a versatile and pre-trained model for various NLP tasks. T5 models are fine-tuned on question generation tasks and can automatically generate meaningful questions that are relevant to the given context.

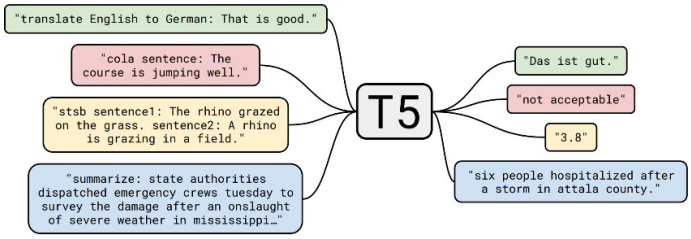



Figure 4.8 T5 question generation model [8]

**4.8 mBERT (Multilingual BERT):** A multilingual model trained on text data from 104 different languages. This model is a cornerstone for cross-lingual NLP tasks, as it possesses the ability to understand and represent text in multiple languages. mBERT's multilingual capabilities make it a valuable asset for cross-lingual question answering, as it can handle questions and documents in diverse languages without the need for language-specific models.

# 4.9 BERT Working

**Overview:**

BERT's self-attention layer is a key component of the Transformer architecture, enabling the model to capture contextual relationships between words in a sequence. Unlike traditional models that process words independently, self-attention allows BERT to consider the entire context of a sequence when encoding each word.

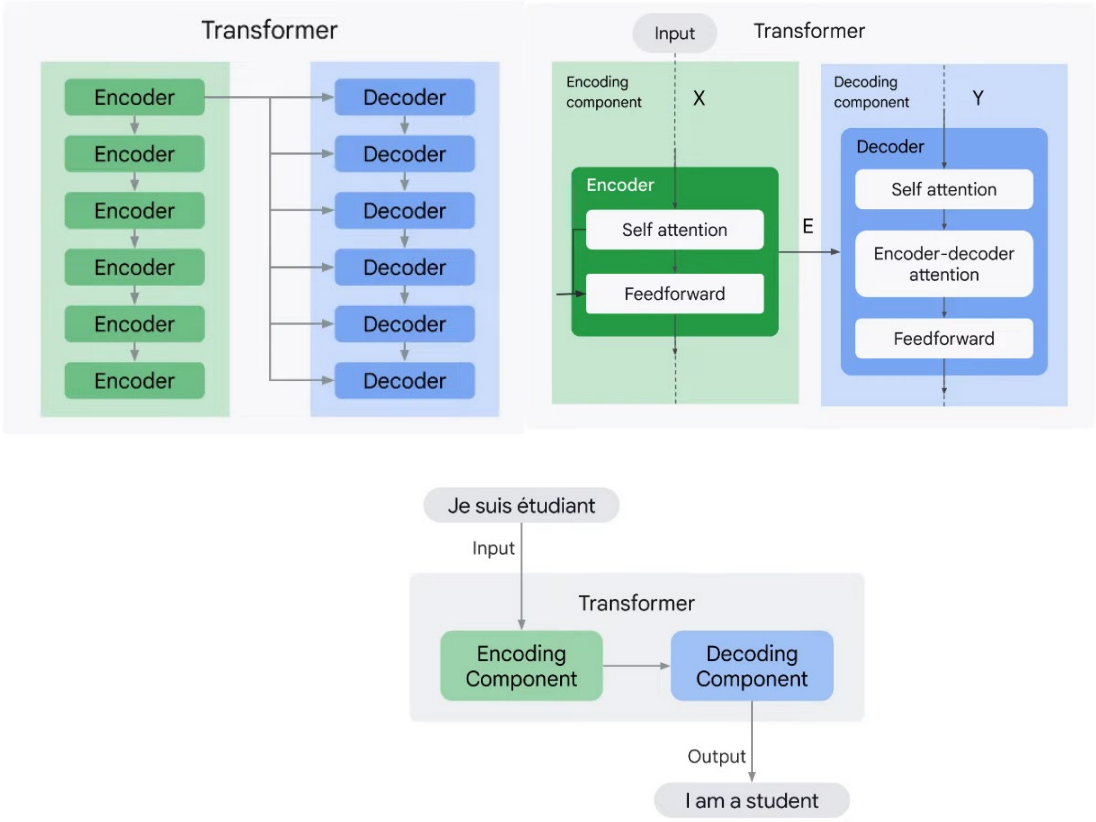



Figure 4.9 Transformer Architecture [6]

### 4.9.1 BERT Transformer Architecture:

- BERT (Bidirectional Encoder Representations from Transformers) is based on the Transformer architecture, introduced in the paper "Attention Is All You Need" in 2017.
- Transformers use self-attention mechanisms to capture relationships between words in a sequence.

- BERT employs a stack of identical encoders, each consisting of a self-attention layer and a feedforward layer.
- The self-attention layer allows the model to focus on relevant parts of the input sequence, capturing context and improving performance in natural language understanding.
- BERT models, specifically Bert Base and Bert Large, differ in the number of layers, with Bert Large having more layers (24 layers) compared to Bert Base (12 layers) as shown in fig. 4.9 and 4.10.

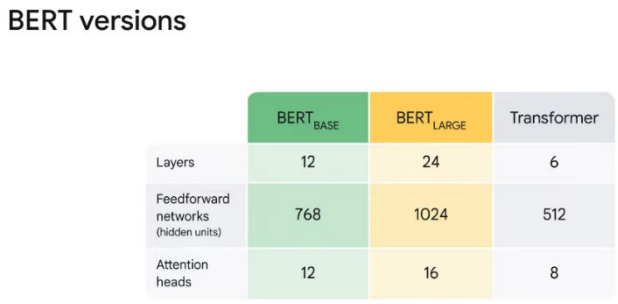



Figure 4.10 BERT Version [6]

### 4.9.2 Pre-training Tasks:

- BERT is pre-trained on various tasks, including a masked language model task where it predicts masked words in a sentence (15% masking recommended).
- Another pre-training task involves predicting the next sentence in a pair, providing the model with a broader context understanding.
- Bert Base has approximately 110 million parameters, while Bert Large has around 340 million parameters.

### 4.9.3 Tokenization:

- Before segmentation, the input text, comprising both the question and the context, undergoes tokenization.
- Tokenization breaks the text into smaller units called tokens, which are the basic processing units for BERT as shown in fig. 4.11.

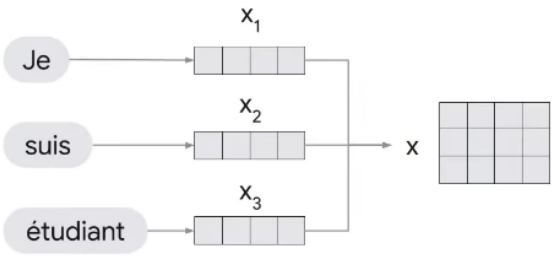



Figure 4.11 Tokenization using BERT [6]

### 4.9.4 Special Tokens:

- BERT uses special tokens to denote different segments in the input sequence.
- The **[CLS]** token is added at the beginning of the sequence, representing the classification or output token.
- The **[SEP]** token is used to separate different segments within the sequence.

### 4.9.5 Input Construction:

- The input sequence is constructed by combining the tokens from both the question and the context.
- It typically follows the pattern: **[CLS] [Question Tokens] [SEP] [Context Tokens] [SEP]**.

### 4.9.6 Segment IDs:

- To enable BERT to distinguish between the question and context, segment IDs are assigned to each token.
- A segment ID of 0 is assigned to tokens corresponding to the **[CLS]**, question, and the first **[SEP]**.
- A segment ID of 1 is assigned to tokens corresponding to the context and the second **[SEP]**.
- These segment embeddings or segmentation vectors that are added to the token embeddings as depicted in fig 4.12.

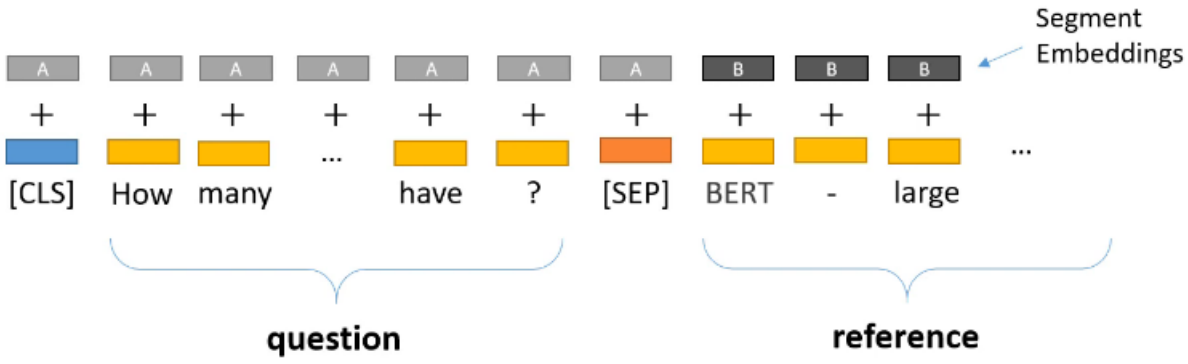



Figure 4.12 BERT Segmentation Embeddings [13]

### 4.9.7 Multi-Head Self-Attention

BERT employs multi-head self-attention mechanisms in each layer. Let's break down how self-attention works:

1. Query, Key, and Value Vectors:
    - For each token in the input sequence, three vectors are derived: Query (Q), Key (K), and Value (V). These vectors are obtained by linear transformations of the input embeddings.
    - These vectors are computed using learned weights during the training process.
    - At this point as dependencies exist between all the input embeddings so they cannot be run in parallel.

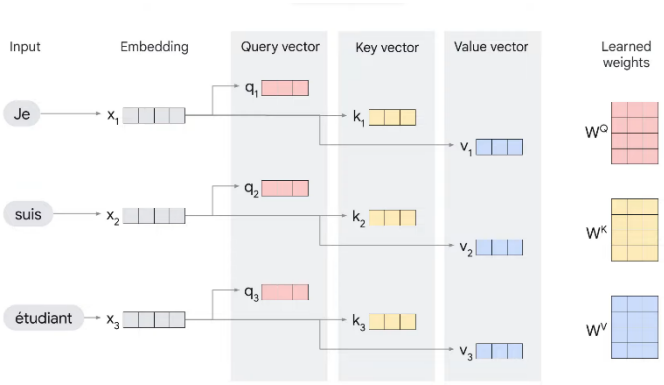



Figure 4.13 Embeddings to QKV vectors [13]

- The QKV weights are multiplied with the input embedding to obtain the QKV vectors as displayed in fig. 4.13 and 4.14.

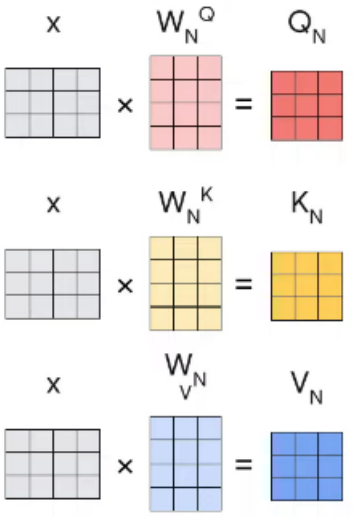



Figure 4.14 Learned weights using QKV vectors [13]

2. Calculating Attention using QKV:

   - The attention scores are computed by taking the dot product of the Query and Key vectors, scaled by the square root of their dimensionality. This process is then followed by a SoftMax operation to obtain attention weights.
   - In essence this layer keeps the values of relevant words intact while it multiplies the values of the insignificant words by very small values so as to make their values less. The mathematical representation is shown in fig. 4.15.

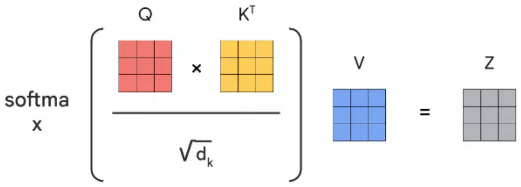



Figure 4.15 Calculating Attention using QKV [13]

3. Multi-Head Attention:

   - BERT employs multi-headed attention, meaning the self-attention mechanism is applied multiple times (e.g., eight heads) in parallel.

- As each head learns different attention patterns, this enables the model to capture different aspects of the context and enhances its ability to represent complex relationships within the input sequence.
- Each attention head produces a final vector of the same dimension of the input embedding as shown in fig. 4.16.

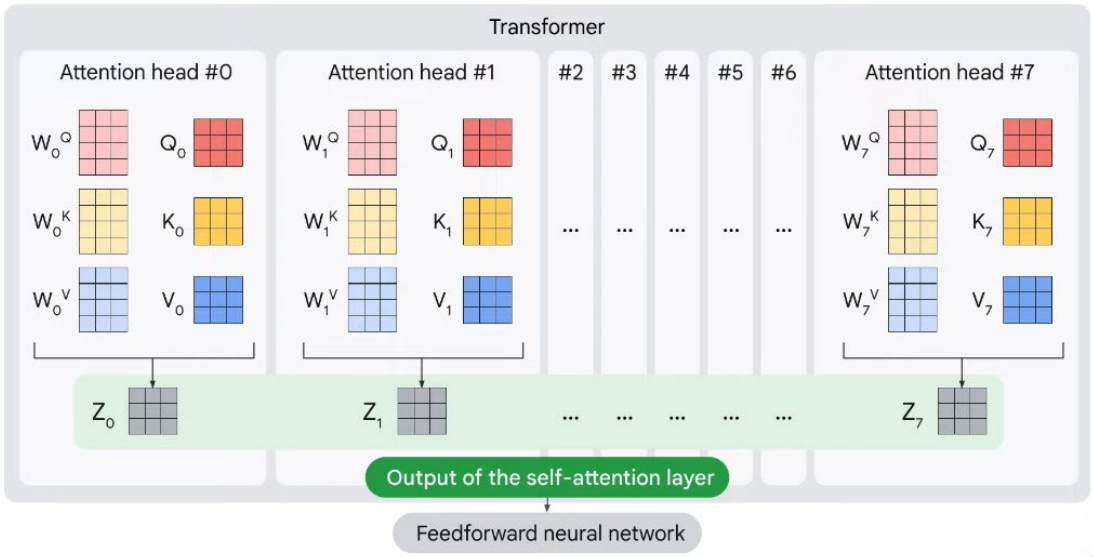



Figure 4.16 Multi-Head Attention layers [1]

4. Concatenate matrix to produce final matrix:
   - The output of each attention head is concatenated and linearly transformed to produce the final output which is the same dimension as the input embedding and is passed to the feed forward layer as shown in fig. 4.17.

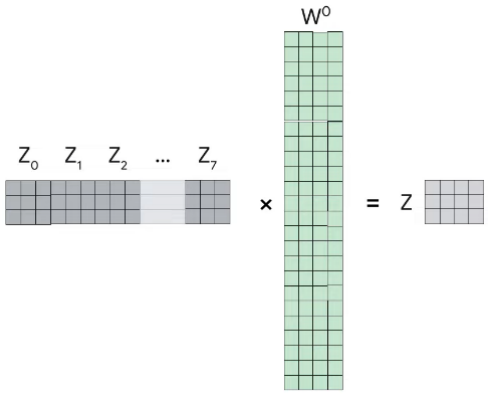


Figure 4.17 Final matrix calculation [1]

### 4.9.8 Feed Forward Layer

After the multi-head self-attention mechanism, the output passes through a feedforward neural network which is independently applied to each position. As The feedforward layer does not have dependencies between paths this allows the various paths to be executed in parallel. This parallelization in the feedforward layer contributes to the efficiency of BERT's architecture.

This network consists of two linear transformations separated by a non-linear activation function (commonly ReLU).

1. Linear Transformation 1:
    - The first linear transformation projects the multi-head self-attention output to an intermediate dimension.
2. Activation Function:
    - An activation function (commonly ReLU) is applied elementwise to introduce non-linearity.
3. Linear Transformation 2:
    - The second linear transformation maps the intermediate representation back to the original dimensionality.
4. Residual Connection and Layer Normalization:
    - A residual connection is added around the feedforward network, followed by layer normalization. This helps stabilize training.

### 4.9.9 Answer Span:

- For question answering tasks, BERT employs start token and end token classifiers to predict the positions of the start and end of the answer span.
- The predictions for start and end positions are obtained by applying the softmax activation function to the dot product of the learned representations and the corresponding weights for start and end tokens.

- The softmax function generates a probability distribution over all the tokens in the sequence, indicating the likelihood of each token being the start or end of the answer span.
- The token with the highest probability for the start position and the token with the highest probability for the start and the end position are selected as the start and end of the answer span, respectively as shown in fig. 4.18.

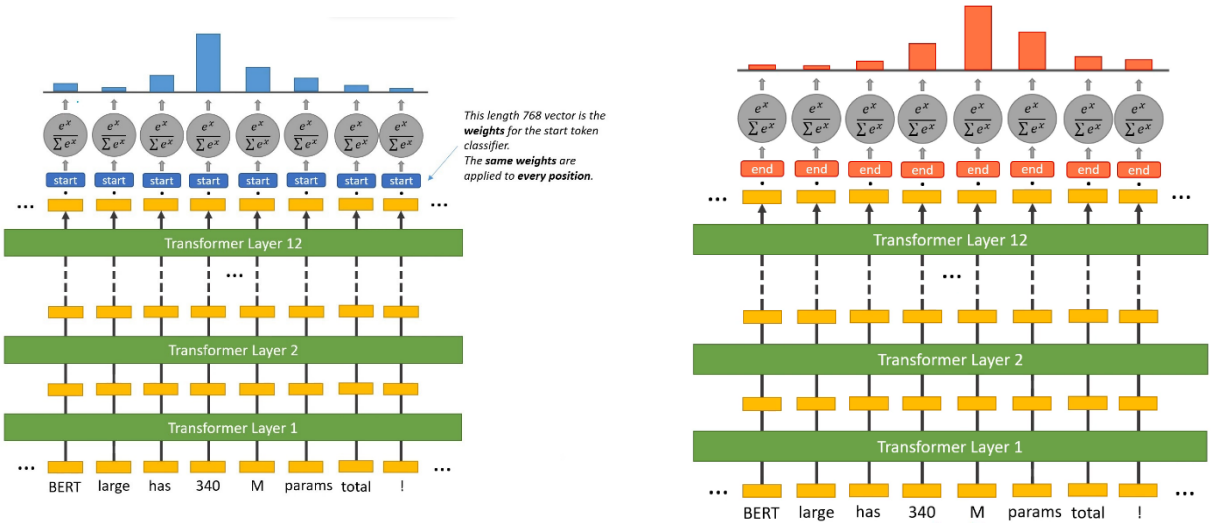



Figure 4.18 Answer span prediction [18]

### 4.9.10 Parallelization:

- One notable feature of the self-attention mechanism is its ability to process input sequences in parallel.
- This parallelization, facilitated by matrix computations, makes BERT computationally efficient and allows it to handle large amounts of data simultaneously.

### 4.9.11 How BERT is used in Question Answering (QA):

BERT's bidirectional nature allows it to consider context from both directions, making it well-suited for question answering tasks. QA involves providing a model with a passage of text and a question, and the model highlights a span of text containing the answer.

**Fine-tuning using Datasets:**

- BERT can be fine tuned using datasets like SQuAD (Stanford Question Answering Dataset), where passages from Wikipedia articles and corresponding questions are used.
- The model is trained to highlight the correct span of text within the reference passage that answers a given question.

**Classification Approach:**

- BERT uses a start token classifier and an end token classifier for identifying the start and end of the answer span.
- The model outputs start and end scores for each token in the sequence.

**Limitations:**

- While powerful, BERT's QA is constrained to identifying answer spans within the provided reference text.

It doesn't generate natural language responses and may not handle questions requiring information synthesis from multiple sentences.

# 4.10 HELSINKI NLP

### 4.10.1 Helsinki-NLP Transformers for Machine Translation:

Helsinki-NLP transformers represent a category of machine learning models designed for translating text across languages. These transformers leverage the Transformer architecture, a powerful deep learning neural network known for its effectiveness in natural language processing (NLP) tasks.


**Fig 4.19** Helsinki-NLP [14]

### 4.10.2 Training Process:

Helsinki-NLP transformers undergo training on extensive parallel text datasets, exposing them to diverse examples in both source and target languages. This training facilitates the learning of intricate patterns and relationships between words in each language, forming the foundation for accurate and context-aware translations.

**Translation Process:**

When employing Helsinki-NLP transformers for text translation, the process involves the following steps:

1. Input Processing: Users provide the model with the text intended for translation.
2. Tokenization: The model tokenizes the input text, breaking it down into individual words or phrases.
3. Vocabulary Lookup: Each token is looked up in the model's vocabulary, generating representations for translation.
4. Translation: Utilizing these representations, the model performs the translation from the source language to the target language.

.

### 4.10.3 Benefits of Helsinki-NLP Transformers:

Helsinki-NLP transformers offer several advantages, making them a preferred choice for machine translation tasks:

1. Accuracy: These transformers demonstrate high accuracy, consistently outperforming traditional machine translation methods in various scenarios.
2. Speed: Known for their efficiency, Helsinki-NLP transformers operate at high speeds, enabling real-time translation of text.
3. Scalability: The transformers are designed to handle translation tasks across a wide array of languages, showcasing scalability.
4. Ease of Use: Helsinki-NLP transformers are user-friendly and easily integrable into diverse applications, contributing to their accessibility.

In summary, Helsinki-NLP transformers emerge as a robust solution for text translation, offering a blend of accuracy, speed, scalability, and ease of use. Their superiority over traditional machine translation methods positions them as a powerful tool in the realm of multilingual communication.

# METHODOLOGY

## 5.1 SDLC Approach (software development lifecycle):

The Spiral Model (refer to fig. 5.1.1) is a risk-driven development methodology that is particularly suitable for projects where there is a high degree of uncertainty, evolving requirements, and continuous reassessment.

- Planning
- Risk Analysis
- Engineering
- Evaluation

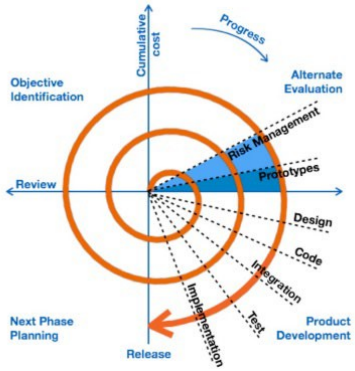



Figure 5.1.1 Spiral model lifecycle [19]

## 5.2 Working on Mandarin Language:

We used “bert-base-chinese” for this, it uses a pretrained BERT model for understanding Chinese text and, when given a question in English and a Chinese document, identifies and extracts the answer from the document, showcasing cross-lingual language understanding and retrieval capabilities.


Figure 5.2.1 Mandarin document snippet

```
In [1]: import torch
        from transformers import AutoTokenizer, AutoModelForQuestionAnswering

In [2]: tokenizer = AutoTokenizer.from_pretrained("bert-base-chinese")
        model = AutoModelForQuestionAnswering.from_pretrained("bert-base-chinese")

        Downloading:   0%|          | 0.00/29.0 [00:00<?, ?B/s]

In [3]: question = "What is the capital of France?"
        document_file = "chinese_document.txt"
```

Figure 5.2.2 Document identification using BERT-base Chinese

## 5.3 Approach: BERT

Our base model was pre-trained on bert-large-uncased-whole-word-masking-finetuned-squad since our first milestone was to develop an accurate answer span prediction algorithm. In this approach we developed a rudimentary answer span prediction algorithm which we tested using small contexts such as paragraphs and queries on the same to identify if the model could predict span accurately.

## 5.4 Approach: BM25 + BERT

By combining the BM25 algorithm for document retrieval and BERT for precise answer extraction this approach retrieved the relevant documents and then extracted accurate answers from those documents without spending extra time running BERT on the other documents where the answer is less likely to be found compared to the previous approach. Here's a detailed breakdown of how we progressed:

### 5.4.1 Document Preprocessing

To answer user queries, we first need access to the text contained within a collection of documents. These documents are typically stored in a folder, and we extract the text from each document. While doing so, we clean the text to remove any extraneous characters and ensure it's in a consistent format. Non-ASCII characters, if present, are handled to avoid issues with text processing.

### 5.4.2 Text Tokenization

Text tokenization by breaking down the text into smaller units. Crucial for BM25 ranking. In this way each document's text is transformed into a list of tokens or words, which allows for more granular analysis of the text.

```
tokenized_text = [nltk.word_tokenize(doc) for doc in text]
```

Figure 5.4.2.1 Text tokenization code snippet

### 5.4.3 Document Retrieval using BM25

Our implementation relies on the utilization of the BM25Okapi algorithm (refer to fig. 5.4.3.1), a variant of the well-established BM25, to discern the ranking of documents based on their applicability to specific user queries. This process plays a crucial role in the identification and ranking documents based on how likely the answer for given query will be in them.

```
bm25 = BM25Okapi(tokenized_text)
```

Figure 5.4.3.1 BM25 culculation code snippet

### 5.4.4 BERT Question Answering Pipeline Initialization

We initialize a question-answering pipeline. This pipeline is responsible for processing queries and extracting answers from the documents. It leverages the BERT model and tokenizer, making the entire process more efficient. This pipeline is fed both the query along with the context. Initially we finetuned BERT on all available documents. But in subsequent iterations, we enhance this operation by implementing a top-k documents ranked by BM25(one at a time) as context to find the answer.

```
Enter Queries

queries = [
    "who are araragi's sisters?",
    "What is the name of the first apparition encountered in Bakemonogatari ?",
    "What school does Koyomi Araragi go to?",
    "Describe the relationship between Araragi and Senjougahara Hitagi"
]

Initialize the question answering pipeline Using BERT (bert-large-uncased-whole-word-masking-finetuned-squad)

qa_pipeline = pipeline("question-answering", model="bert-large-uncased-whole-word-masking-finetuned-squad")

All PyTorch model weights were used when initializing TFBertForQuestionAnswering.

All the weights of TFBertForQuestionAnswering were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertForQuesti
```

Figure 5.4.4.1 QA-pipeline initialization code snippet

### 5.4.5 Question Answering and Results

We executed the code above for various sets of documents and noted that the results tend to be better if the documents and queries pertain to a homogeneous subject instead of containing a wide range of diverse subjects.

### 5.4.6 Conclusion and Output

In the final part of this approach, we present the results of the question-answering process. Each query's results are displayed.

```
for query, answer_span, document_title in query_results:
    print("\nQuery:", query)
    print("Document Title:", document_title)
    print("Answer Span:", answer_span)
    print("-" * 100)
```

```
Query: who are araragi's sisters?
Document Title: BAKEMONOGATARI.txt
Answer Span: sistersKaren andTsukihi
----------------------------------------------------------------------------------------------------

Query: What is the name of the first apparition encountered in Bakemonogatari ?
Document Title: KABUKIMONOGATARI.txt
Answer Span: Hanekawa
----------------------------------------------------------------------------------------------------

Query: What school does Koyomi Araragi go to?
Document Title: KABUKIMONOGATARI.txt
Answer Span: Hachikuji
----------------------------------------------------------------------------------------------------

Query: Describe the relationship between Araragi and Senjougahara Hitagi
Document Title: NEKOMONOKATARI-_WHITE_.txt
Answer Span: Broadened horizons
----------------------------------------------------------------------------------------------------
```

Figure 5.4.6.1 Document wise answer span results

## 5.5 Approach: BERT Model Caching

Additionally, we considered catching the BERT model. If the BERT model hasn't already been cached (downloaded and stored for future use), it's fetched from a model repository, cached, and ready for use. As we believed that caching would save time and resources, especially when working with large language models like BERT. But this method did not significantly impact the execution time as we still had to fine tune the BERT model each time on the document the alternative being caching the weights after training the model on each document which would require a lot of storage space and was deemed infeasible.

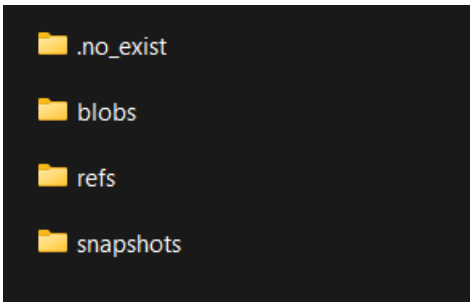



**Figure 5.5.1** Project file structure

### 5.5.1 Results and observations:

We executed the model on two types of documents.

1. Novels (over 500 pages) – A series of novels and their sequels having similar contexts.
2. User manuals (40 – 50 pages) – Unrelated instruction manuals for different appliances for a diverse pool of information. Refer to fig. 5.5.1.1 for more examples.

```
--------------------------------------------------------------------------------
Document Title: amafit trex pro usermanual.pdf
Answer Span: Flashlight
Tap the flashlight icon to enable the flashlight
--------------------------------------------------------------------------------
Document Title: Range-Rover-Brochure-1L4602200000BINEN01P.pdf
Answer Span: Electronic Air Suspension with Dynamic Response Pro
--------------------------------------------------------------------------------
Total processing time: 9233.23957824707 seconds
```

Fig 5.5.2 Result for User Manual

We observed that the user manuals, due to having shorter context and not having to give BERT a context of over 500 pages had more accurate results prompting us to think about how to improve our model for larger documents.

## 5.6 QA Generation and Model Fine Tuning

Building upon the previous code versions, we implemented an approach that leverages the provided documents to generate question and answer pairs. Leveraging these QA pairs to then fine-tune the BERT model within the same framework. The following section outlines the execution of this approach:

### 5.6.1 Proporcessing the documents

We set the path to the folder containing the documents that we want to process. The documents are loaded from this folder, and their text content is extracted. During this process, we ensure that the text is cleaned and formatted consistently to facilitate further analysis. We store the cleaned and formatted documents separately within txt files such that the same step won't have to be repeated for already cleaned documents.

### 5.6.2 Initializing the T5 Model and Question Generation

We begin by initializing the T5 model which is responsible for generating QA pairs based on the provided context and specify a cache directory. We create questions and answers in a structured JSON format which is required by BERT to fine tune it on the same. This was performed using the code shown in fig. 5.6.1.

```
def initialize_t5_model(model_name, cache_dir):
    tokenizer = AutoTokenizer.from_pretrained(model_name, cache_dir=cache_dir)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name, cache_dir=cache_dir)
    return tokenizer, model
```

**Fig 5.6.1** T5 Model Pipeline

**5.6.2.1 Caching Model Weights and Structuring**: QA Data To optimize performance, we define a path for caching model weights. The model weights for the T5 model are cached to ensure efficient processing in subsequent steps.

**5.6.2.2 Structuring QA Data:** We structure the data in a desired question-answering format. This format includes paragraphs and QA pairs for each document. The questions and answers generated by the T5 model are organized and associated with the corresponding document and it's paragraphs from which they were generated, they were structure as given below.

- Title of the document.
- Paragraph
- Question
- Answer

### 5.6.2.3 Saving QA data to JSON

The QA data, organized as structured JSON data, is saved to a JSON file. This file (refer to fig.5.6.2) will be used in the subsequent steps of the process.

```
# Save the QA data to a JSON file
output_json_file = 'qa_data.json'
with open(output_json_file, 'w', encoding='utf-8') as json_file:
    json.dump(qa_data, json_file, ensure_ascii=False, indent=4)
```

**Fig 5.6.2** Saving data in JSON format

### 5.6.3 Fine-tuning BERT Model Training

This approach implements a comprehensive pipeline for generating questions using the T5 model, structuring QA data, fine-tuning a BERT model for question answering, and ultimately creating a system capable of answering questions based on the generated questions and answers. This approach uses both T5 and BERT models to optimize the performance of the question answering system.

## 5.7 Fine Tuning on SQuAD Dataset

After pretraining, the BERT model underwent fine-tuning on the SQuAD (Stanford Question Answering Dataset) for question-answering tasks.

Fine-tuning Process: Dataset Preparation: Obtain the SQuAD dataset, which consists of question-answer pairs related to a given context. Each example includes a context passage, a question, and the corresponding answer span in the context.

Tokenization: Tokenize the SQuAD dataset using the BERT tokenizer, ensuring that the input format aligns with BERT's expectations.

Model Configuration: Load the pretrained BERT model and configure it for question-answering tasks using the code in fig. 5.7.

```
In [ ]: from transformers import BertForQuestionAnswering, BertTokenizer

model = BertForQuestionAnswering.from_pretrained('bert-base-uncased')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

optimizer = torch.optim.Adam(model.parameters(), lr=2e-5)
loss_fn = torch.nn.CrossEntropyLoss()

# Fine-tuning loop
for epoch in range(num_epochs):
    for batch in train_dataloader:
        inputs = batch['input_ids']
        labels = batch['start_positions']
        outputs = model(inputs)
        loss = loss_fn(outputs.start_logits, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

```
    optim.step()

    loop.set_description(f'Epoch {epoch+1}')
    loop.set_postfix(loss=loss.item())

/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:411: FutureWarning
  warnings.warn(
Epoch 1: 100%|██████████| 5427/5427 [2:18:46<00:00,  1.53s/it, loss=1.46]
Epoch 2: 100%|██████████| 5427/5427 [2:18:09<00:00,  1.53s/it, loss=1.73]
Epoch 3:  60%|██████    | 3265/5427 [1:22:27<54:20,  1.51s/it, loss=0.539]
```

**Fig 5.7** Model training

Evaluation: Regularly evaluate the fine-tuned model on a validation set to monitor performance. Save the Fine-tuned Model: Save the fine-tuned model for future use or deployment.

This summarizes the steps involved in fine-tuning the pretrained BERT model specifically for question-answering tasks on the SQuAD dataset. Adjustments may be required based on specific implementation details and requirements.

## 5.8 Building an in-hand translator.

### 5.8.1 Seamless Communication Across Languages:

The primary objective of the in-hand translator is to enable smooth and efficient communication, breaking down language barriers. It caters to a diverse user base by supporting languages like Vietnamese, Chinese, Arabic, German, and English.

### 5.8.2 Helsinki NLP Model for Multilingual Translation:

The translator integrates the Helsinki Natural Language Processing -NLP model, a robust transformer-based language model. This incorporation enhances the translation function's efficiency, especially when dealing with multilingual contexts.

### 5.8.3 User-Friendly Cross-Lingual Model:

The translator's utility is amplified when used in conjunction with a Cross Lingual model. It autonomously identifies the language of the input query and seamlessly translates it into the user's selected language. This user-friendly approach simplifies the translation process, making it convenient for users to copy and utilize the translated text as needed.

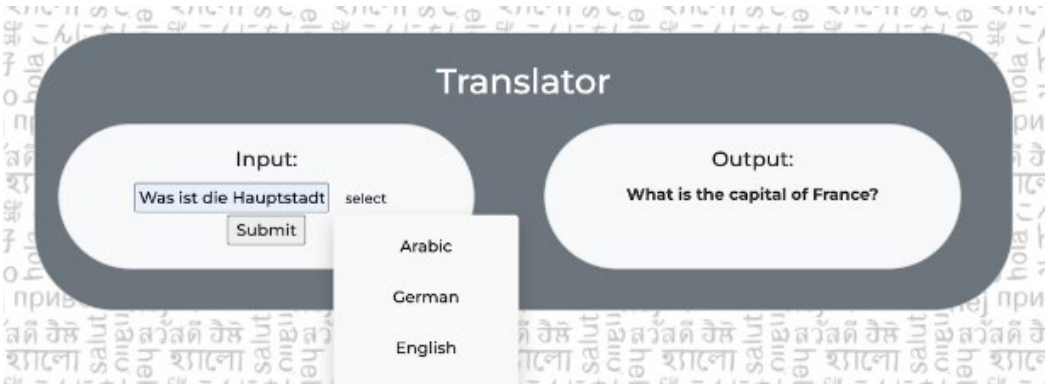



**Fig 5.8.1** In hand translator

## 5.9 Approach: BM25 + BERT + Cross Lingual

### 5.9.1 Query Translation:

The initial step involves leveraging our in-hand translator to seamlessly translate user queries into five languages: English, Chinese, German, Vietnamese, and Arabic. This ensures a comprehensive search across diverse linguistic contexts.

### 5.9.2 BM25 Search in Each Language:

Employ the BM25 algorithm, a robust information retrieval technique, to conduct searches within each language's corpus using the translated queries. BM25 provides a ranked list of relevant documents based on their relevance to the query.

### 5.9.3 Document Retrieval:

Retrieve the top documents in each language based on BM25 scores. These documents serve as candidates for subsequent analysis, ensuring that the system focuses on the most relevant content.

### 5.9.4 BERT Question Answering (QA) Model:

Integrate language-specific BERT-based QA models into the framework. Tailor each QA model to its respective language (English, Chinese, German, Vietnamese, and Arabic) to maximize contextual understanding and extraction accuracy. This ensures the system is optimized for extracting precise answers from documents in each specific language.

### 5.9.5 Answer Extraction:

Apply the BERT QA model to the top documents in each language to extract detailed answers to the user's query. This step ensures that the system delivers accurate and contextually relevant responses.

### 5.9.6 Ranking and Aggregation:

Rank the extracted answers based on relevance or confidence scores from the BERT QA model. Aggregate the top answers from each language to present a comprehensive set of responses to the user's query.

These steps collectively form the architecture of this cross-lingual model as shown in fig 5.9.1, leveraging techniques in information retrieval and natural language processing for effective multilingual query handling.

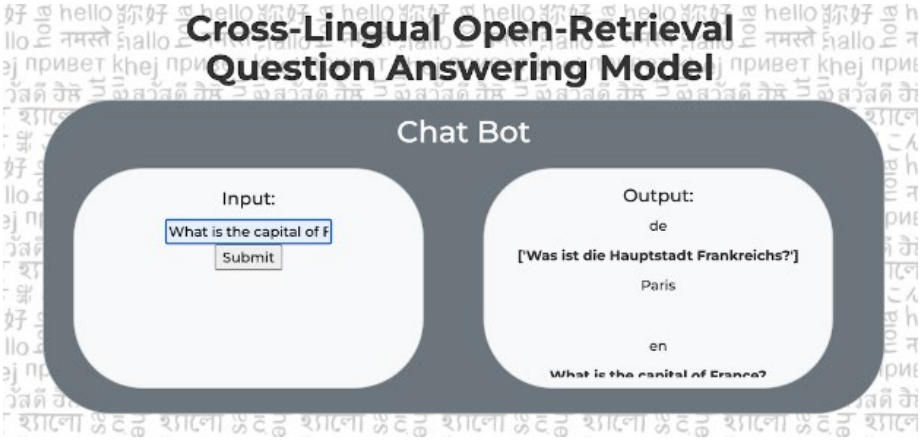



**Fig 5.9.1** Cross lingual model

# CONCLUSION AND RESULT

The evaluation process involves a **two-step approach.**

1. A **T5, a text-to-text transformer**, is employed to generate question-answer pairs based on the provided text.
2. The question-answering mechanism in the project is utilized to determine answers for each generated question. This mechanism is designed to extract relevant information from the input text.

The cross-lingual question-answering (QA) model has been evaluated using a set of 100 questions, with 20 questions in each of the four languages: German (de), English (en), Chinese (zh), and Vietnamese (vi).

The model has been intentionally not provided any QA pairs with answers in the Arabic (ar) documents, but it has still been evaluated on **a total of 100 questions to demonstrate** the comparatively low score showing its good performance for detecting true negatives.

## 6.1 Precision, Recall and F1 score.

Precision, recall, and F1 scores were calculated for each language based on the model's performance on the 100 questions. Given that none of the answers to the 100 questions, except for a few exceptions, were present in the model's training data.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

**Fig 6.1 Precision and Recall [27]**

True Positives (TP): Number of samples correctly predicted as “positive.”
False Positives (FP): Number of samples wrongly predicted as “positive.”
True Negatives (TN): Number of samples correctly predicted as “negative.”
False Negatives (FN): Number of samples wrongly predicted as “negative.”

**F1 Score:** The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, making it a useful metric when there is an uneven class distribution.

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

**Fig 6.2 F1 Score [27]**

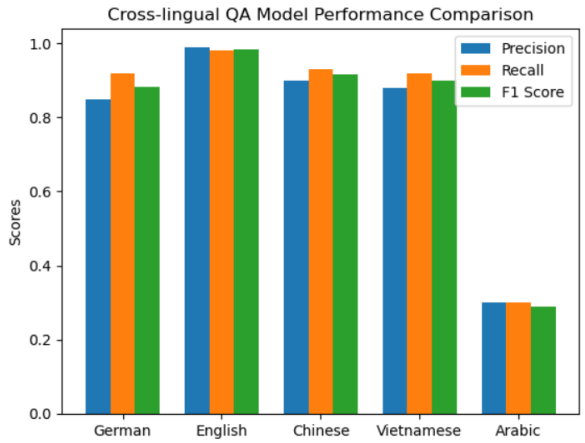



Fig 6.3 Bar graph of precision, recall and F1 score for all 5 languages.

**Interpretation**:

The results reveal that all languages (German, English, Chinese, Vietnamese) achieved high precision, recall, and F1 scores across all three metrics as shown by the bar graph plotted in fig. 6.3.

The comparatively low score for Arabic aligns with expectations, given that the model was not provided answers for any of its questions during training, emphasizing the **importance of having relevant training data for optimal model performance.**

## 6.2 Semantic Similarity Metric: DistilUSE

To assess the semantic similarity between the generated and actual answers, we leverage **the distiluse-base-multilingual-cased-v2** model. This multilingual sentence embedding model converts both the generated and actual answers into vectors.

### 6.2.1 Sentence Embeddings and Mean-Pooling

The sentence embeddings of both the generated and actual answers are created using distiluse-base-multilingual-cased-v2. These embeddings are then mean-pooled to generate representative vectors for each answer.

### 6.2.2 Cosine Similarity Computation:

The similarity score between the vectors is computed using cosine similarity. This metric provides a quantitative measure of the semantic similarity between the generated and actual answers.

By employing this comprehensive evaluation metric, we ensure a nuanced assessment of our question-answering system's performance, considering both the question generation and answer extraction aspects, while leveraging advanced techniques in semantic similarity measurement as shown in fig. 6.1.

```
Language: zh
Q: What is the name of the Buddha statues in Afghanistan?
A:  Bamiyan
Language: de
Language: en

Question: What is the name of the Buddha statues in Afghanistan?
Answer: bamiyan

Similarity scores: 0.9198797345161438
```

**Fig 6.4** Sample similarity score for a query

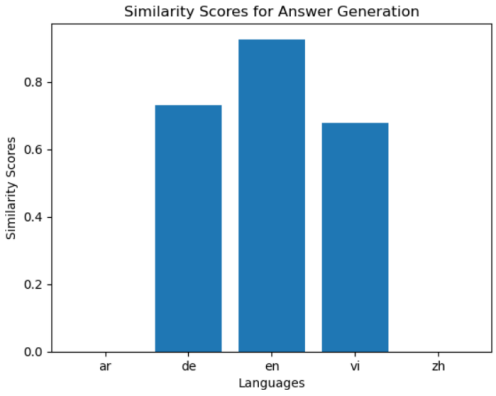



**Fig 6.5** Bar graph to compare scores among languages.

### 6.2.3 Average Similarity Score:

Values in the **Table 6.1** represent the average similarity scores for 20 questions in each language (except Arabic). Scores range from 0.0 (low similarity) to 1.0 (high similarity).

**Average (Total):** The "Average (Total)" row shows the average similarity scores across all languages.

| Languages | de | en | zh | vi | ar | Average |
|---|---|---|---|---|---|---|
| German (de) | 0.92 | 0.36 | 0.35 | 0.33 | 0.27 | 0.446 |
| English (en) | 0.37 | 0.98 | 0.39 | 0.37 | 0.29 | 0.48 |
| Chinese (zh) | 0.36 | 0.39 | 0.95 | 0.4 | 0.28 | 0.476 |
| Vietnamese (vi) | 0.34 | 0.37 | 0.4 | 0.92 | 0.27 | 0.46 |
| Arabic (ar) | 0.28 | 0.29 | 0.28 | 0.27 | 0.29 | 0.282 |
| Average (Total) | 0.454 | 0.478 | 0.474 | 0.458 | 0.28 | |

Table 6.1 Similarity scores for all languages

A heatmap was created based on the data in Table 6.1, using varying shades of colors to represent differences in the corresponding similarity scores.

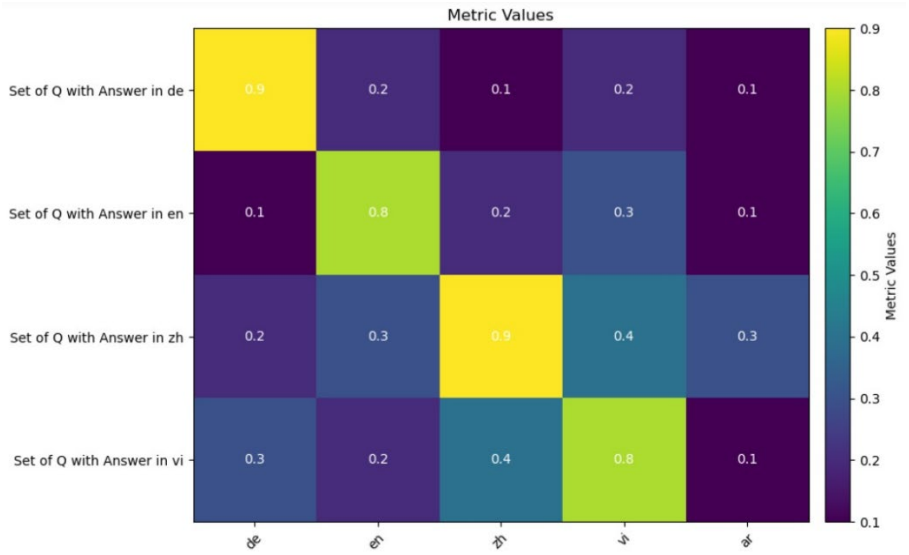



Fig 6.6 Heatmap for Table 6.1 for better understanding

Table **6.1** provides an overview of the average similarity scores between questions and documents in each language. **German, English, Chinese, and Vietnamese languages show relatively higher average similarity scores**, indicating better matches. Arabic language has a lower average similarity score, suggesting lower overall similarity with the Arabic language documents. The "Average (Total)" row provides a summary of the overall performance across all languages.

# PROJECT UI

The integration of the web interface with the back-end software using the Flask framework indicates a cohesive and efficient way to deploy and interact with the question answering system. Let's break down the key components and functionalities mentioned:

1. **Flask Framework Integration:**

   **Web Interface:** Flask is a micro web framework in Python, and its integration suggests a lightweight and scalable solution for serving web pages and managing interactions between the front end and back end.

2. **Translator Functionality:**

   **Translation Service**: The application incorporates a translator feature that allows users to convert answers from the question answering system into a language of their choice. This feature enhances user accessibility and provides a personalized experience as displayed in fig 7.1.

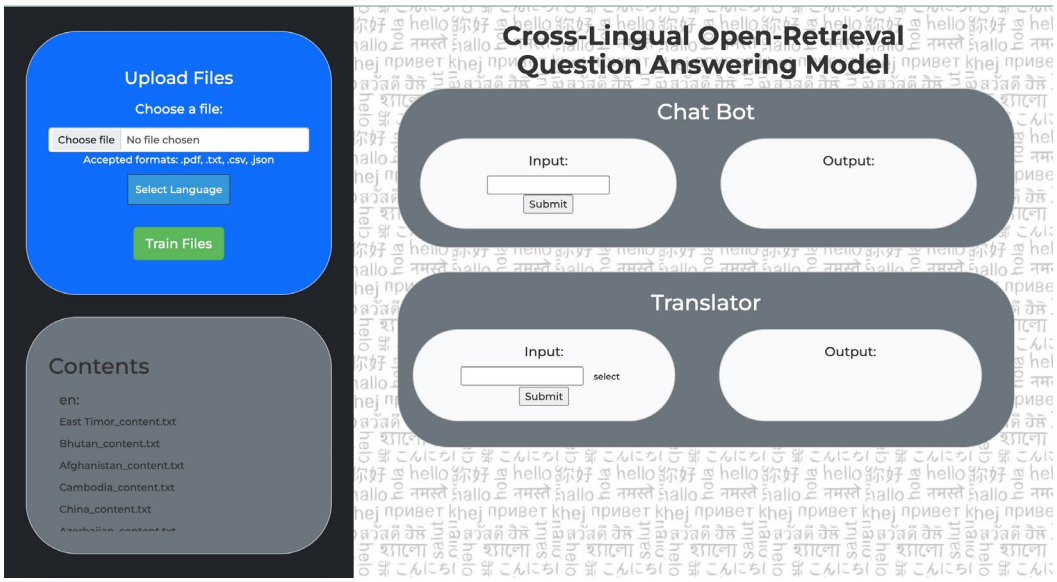



**Fig 7.1** Front page of the Cross Lingual Model withal the functionalities

3. **Custom File Upload Functionality:**

    - **User-Driven Database Expansion:** Users can upload relevant files, allowing them to contribute and expand the database according to their needs, as shown in fig 7.2. This feature empowers users to customize and enhance the system's knowledge base based on their requirements.

4. **Transparent Database Representation:**

    - **Language-Specific File Table:** The application displays a table that categorizes files according to their language, promoting transparency. This table provides users with a clear overview of the available content in the database, organized by language.

5. **Dynamic HTML Pages with Flask:**

    - **Live Updates:** The Python Flask framework is utilized to dynamically generate HTML pages. This approach ensures that the web pages are updated in real-time, reflecting changes in Python variables. This live updating feature enhances the user experience, providing an interactive and responsive interface.

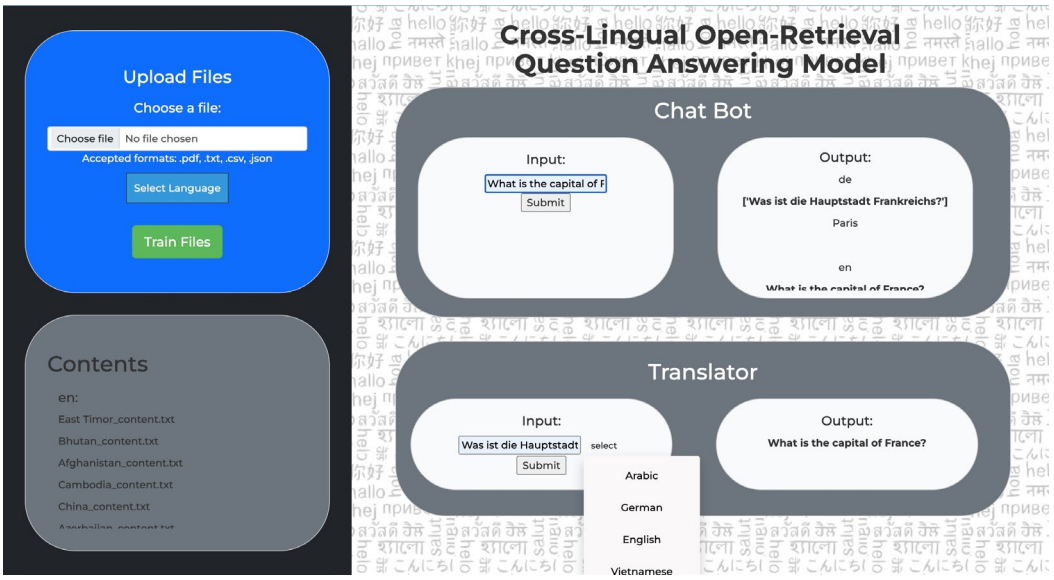



**Fig 7.2** A query is entered to get the result in different languages.

# FUTURE SCOPE

### 8.1 Advanced Content Understanding:

Integrate entity recognition and linking techniques within documents to identify and connect entities like names and locations. This enhancement enriches the answer generation process with contextually relevant information, enhancing the system's understanding of complex content.

### 8.2 Personalized User Interaction:

Introduce personalized user profiles that store preferences, query history, and language preferences. This approach tailors the user experience, providing customized interactions based on individual preferences and past interactions with the system.

### 8.3 Entity Recognition and Linking:

Employ advanced named entity recognition and linking techniques within documents. This augmentation involves identifying and connecting entities such as names and locations, thereby enriching the answer generation process with contextually relevant information, ultimately enhancing the system's understanding of complex content.

### 8.4 Incorporating Zero Shot Cross-Lingual Transfer and Knowledge Base Comprehension:

Merge a pretrained multilingual model like XLM-R with zero-shot transfer learning for cross-lingual question answering. This innovative approach involves leveraging a knowledge base for reading comprehension, enabling the model to generate answers in a target language based on questions in a source language. This strategic integration aims to expand the system's capabilities to answer questions in languages it has never been directly trained on, showcasing adaptability and generalization.

# REFERENCES


1. Alfonso, A. (2023, September 8). Google Cloud Skills Boost — Part 7— Transformer Models and BERT Model: Overview. *Medium*. https://medium.com/google-cloud/google-cloud-skills-boost-part-7-transformer-models-and-bert-model-overview-c63bfe859e5

2. apple. (n.d.). *GitHub - apple/ml-mkqa: We introduce MKQA, an open-domain question answering evaluation set comprising 10k question-answer pairs aligned across 26 typologically diverse languages (260k question-answer pairs in total). The goal of this dataset is to provide a challenging benchmark for question answering quality across a wide set of languages. Please refer to our paper for details, MKQA: A Linguistically Diverse Benchmark for Multilingual Open Domain Question Answering*. GitHub. https://github.com/apple/ml-mkqa/

3. Artetxe, M., Ruder, S., & Yogatama, D. (2020). On the Cross-lingual Transferability of Monolingual Representations. *On The Cross-lingual Transferability of Monolingual Representations*. https://doi.org/10.18653/v1/2020.acl-main.421

4. Asai, A., Kasai, J., Clark, J. H., Lee, K., Choi, E., & Hajishirzi, H. (2021). XOR QA: Cross-lingual Open-Retrieval Question Answering. *XOR QA: Cross-lingual Open-Retrieval Question Answering*. https://doi.org/10.18653/v1/2021.naacl-main.46

5. *bert-base-chinese · Hugging Face*. (n.d.). https://huggingface.co/bert-base-chinese

6. Bhaskar, Y. (2023, June 23). Google Cloud Skill Boost: Gen AI Course Notes : Lecture 8: Transformer Models and BERT Model. *Medium*. https://medium.com/@yash9439/google-cloud-skill-boost-gen-ai-course-notes-

lecture-8-transformer-models-and-bert-model-66b3a9ae09fb

7. *bhavikardeshna/xlm-roberta-base-vietnamese · Hugging Face*. (n.d.). https://huggingface.co/bhavikardeshna/xlm-roberta-base-vietnamese

8. Chen, G., Ma, S., Chen, Y., Dong, L., Zhang, D., Pan, J., Wang, W., & Wei, F. (2021). Zero-Shot Cross-Lingual Transfer of Neural Machine Translation with Multilingual Pretrained Encoders. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/2021.emnlp-main.2

9. Chen, Q. (2021, December 14). T5: a detailed explanation - Analytics Vidhya - Medium. *Medium*. https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51

10. Chu, L. (2023, July 2). Understanding Term-Based retrieval Methods in information retrieval. *Medium*. https://towardsdatascience.com/understanding-term-based-retrieval-methods-in-information-retrieval-2be5eb3dde9f

11. *deepset/bert-base-uncased-squad2 · Hugging Face*. (2001, April 5). https://huggingface.co/deepset/bert-base-uncased-squad2

12. *deutsche-telekom/bert-multi-english-german-squad2 · Hugging Face*. (n.d.). https://huggingface.co/deutsche-telekom/bert-multi-english-german-squad2

13. *Figure 1: Example of input modification to fit the QA paradigm for a. . .* (n.d.). ResearchGate. https://www.researchgate.net/figure/Example-of-input-modification-to-fit-the-QA-paradigm-for-a-sentence-that-contains-an_fig1_350875656

14. *Helsinki-NLP (Language Technology Research Group at the University of Helsinki)*. (n.d.). https://huggingface.co/Helsinki-NLP

15. *Helsinki-NLP/Opus-MT: Open neural machine translation models based on Marian-NMT*. (2022, October 23). LibreTranslate Community. https://community.libretranslate.com/t/helsinki-nlp-opus-mt-open-neural-machine-translation-models-based-on-marian-nmt/371

16. Paulmelki. (n.d.). *GitHub - paulmelki/BERT_BM25_InformationRetrieval: This project aims at creating a search engine based on BERT language model.* GitHub. https://github.com/paulmelki/BERT_BM25_InformationRetrieval

17. *sentence-transformers/distiluse-base-multilingual-cased-v2 · Hugging Face*. (n.d.). https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2

18. Singh, R. K. (2021, April 9). *How to Train A Question-Answering Machine Learning Model | Paperspace Blog*. Paperspace Blog. https://blog.paperspace.com/how-to-train-question-answering-machine-learning-models/

19. SumatoSoft. (2022, December 27). How to choose the best software development model for your project. *Medium*. https://sumatosoft.medium.com/how-to-choose-the-best-software-development-model-for-your-project-244692df7b7c

20. *Text Cookbook*. (n.d.). TensorFlow. https://www.tensorflow.org/hub/tutorials/text_cookbook

21. *The advantage of Self-Supervised Learning*. (n.d.). https://www.lightly.ai/post/the-advantage-of-self-supervised-learning

22. *Universal Sentence Encoder*. (n.d.). TensorFlow. https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder

23. Valdez, R. a. C. (2023, February 4). Building an Amazon Prime content-based Movie Recommender System. *Medium*. https://medium.com/geekculture/building-an-amazon-prime-content-based-movie-recommendation-system-d4bdc35d8bbf

24. *XOR-TYDi QA*. (n.d.). https://nlp.cs.washington.edu/xorqa/

25. Zhang, C. (2023, February 26). *Cross-Lingual Question Answering over Knowledge Base as Reading Comprehension*. arXiv.org. https://arxiv.org/abs/2302.13241

26. Kundu, R. (2023, April 20). F1 Score in Machine Learning: Intro & Calculation. *V7*. https://www.v7labs.com/blog/f1-score-guide

Date: 09/12/23

Signature of Group Member(s), Date

Signature of Project Advisor, Date