Low Resource Question Answering: An Amharic Benchmarking Dataset

Tilahun Abedissa Taffa[1,2,3], **Yaregal Assabie**[2], **and Ricardo Usbeck**[3]
[1]Semantic Systems, Universität Hamburg, Hamburg, Germany
[2]Department of Computer Science, Addis Ababa University, Addis Ababa, Ethiopia
[3]Leuphana Universität Lüneburg, Lüneburg, Germany
tilahun.taffa@uni-hamburg.de, yaregal.assabie@aau.edu.et, ricardo.usbeck@leuphana.de

Abstract

Question Answering (QA) systems return concise answers or answer lists based on natural language text, which uses a given context document. Many resources go into curating QA datasets to advance the development of robust QA models. There is a surge in QA datasets for languages such as English; this is different for low-resource languages like Amharic. Indeed, there is no published or publicly available Amharic QA dataset. Hence, to foster further research in low-resource QA, we present the first publicly available benchmarking **Amh**aric **Qu**estion **A**nswering **D**ataset (Amh-QuAD). We crowdsource 2,628 question-answer pairs from over 378 Amharic Wikipedia articles. Using the training set, we fine-tune an XLM-R-based language model and introduce a new reader model. Leveraging our newly fine-tuned reader run a baseline model to spark open-domain Amharic QA research interest. The best-performing baseline QA achieves an F-score of 80.3 and 81.34 in retriever-reader and reading comprehension settings.

Keywords: Low Resource Question Answering, Amharic Question Answering Dataset, Amharic Reading Comprehension, Amh-QuAD

1. Introduction

The task of Question Answering (QA) is to accurately retrieve an answer to a natural language question from a certain underlying data source (Chen and Yih, 2020). The standard train & test QA dataset creation is applied to evaluate models' question synthesis ability and answer accuracy. Crowdsourcing or automatic generation are common approaches in curating QA datasets (Dzendzik et al., 2021). In the crowdsourcing approach, crowd-workers formulate question-answer pairs within a given context. Crowdsourcing allows for the creation of high-quality question-answer pairs, but it is expensive. In contrast, automatic generation approaches leverage language generation models, templates, or machine translation in formulating question-answer pairs. However, attaining a reliable model capable of generating question-answer pairs as accurate as those from a human poses a challenge. Therefore, studies introduce humans in the loop to minimize the generation of trivial, un-grammatical, and incorrect question-answer pairs (Cambazoglu et al., 2020; Fabbri et al., 2020).

The distinction between the existing QA datasets lies in 1) the question expected answer: factoid vs. non-factoid, 2) the data source domain: closed vs. open, and 3) the answer formulation sub-task: extractive vs. generative. Factoid questions like "Who is the founder of Ethiopia's capital Addis Ababa?" (Answer: "Emperor Menelik II") requires a named entity such as proper noun, date, number, or short phrase as an answer (Abedissa and Libsie, 2019).

Context: …በላሊበላ 11 ውቅር ዐብያተ ክርስቲያናት ያሉ ሲሆን ከነዚህም ውስጥ ቤተ ጊዮርጊስ (ባለ መስቀል ቅርፅ) ሊታይ ውሃ ልኩን የጠበቀ ይመስላል። ቤተ መድኃኔዓለም የተባለው ደግሞ ከሁሉም ትልቁ ነው። ላሊበላ (ዳግማዊ ኢየሩሳሌም) የገና በዓል ታህሳስ 29 በልዩ ሁኔታና ድምቀት ይከበራል፣ "ቤዛ ኩሉ" ተብሎ የሚጠራው በነግህ የሚደረገው ዝማሬ በዚሁ በዓል የሚታይ ልዩና ታላቅ ትዕይንት ነው።
(While there are 11 rock-hewn churches in Lalibela, of these churches, *betə giorgis* 'House of St. George' (the one that is cross-shaped) appears to have a leveled foundational platform. The church named *betə medhanialəm* (House of the Saviour of the World), is also the biggest of all. In Lalibela (the Second Jerusalem), *gənna* 'Christmas' holiday is celebrated uniquely and colorfully on December 29. The song called *beza kulu* is played in the aftermath of the holiday and it is a great and special scene observed in this holiday.)
Question: ከላሊበላ አስራ አንዱ ውቅር አብያተ ክርስቲያናት የመስቀል ቅርጽ ያለው የትኛው ነው? (Of the 11 Lalibela's rock-hewn churches, which one is cross-shaped?)
Answer: ቤተ ጊዮርጊስ (*betə giorgis* 'House of St. George')

Figure 1: Amh-QuAD context, question, and answer triplets.

Unlike that, how, why, opinion, definition, and recommendation questions fall into the non-factoid category. For example, a question like "Why does water appear colorless and tasteless?" compels gathering relevant information, reasoning, and synthesizing multiple information pieces from different sources (Yang et al., 2019). Hence, based on the question types, a QA model and its benchmarking dataset are factoid or non-factoid (Dzendzik

The Fifth Workshop on Resources for African Indigenous Languages @LREC-COLING-2024 (RAIL), pages 124–132

et al., 2021). Besides, the data source used to answer a question contains generic information about many things or information specific to a particular domain, like sports, geography, or medicine. Thus, based on the domain of the data source and the question, domain-dependent QA systems are referred to as closed and domain-independent as open QA (Chen and Yih, 2020). Furthermore, QA datasets and models differ in how the answer is retrieved - extractive or generative. Extractive QA datasets like SQuAD (Rajpurkar et al., 2016) measure a QA model competency in predicting the corresponding start and end tokens of the answer span from a context. Unlike that, generative QA datasets contain questions whose answer is a context comprehension, not a direct copy (Raffel et al., 2020).

The architecture of QA systems typically includes question analysis modules to understand questions, information retrieval (IR) systems to locate relevant documents or data and answer extraction mechanisms to extract accurate answers from the retrieved information (Abedissa and Libsie, 2019). In which a natural language question comes into the question analysis module, and an answer flows out of the answer processing module (Chen and Yih, 2020). The question analysis component analyzes the input question in several ways. One is a morphosyntactic analysis, assigning the part-of-speech tag to each word in the question, indicating whether a word is a verb, noun, or adjective. Then, classify questions to identify the semantic type of the question (Utomo et al., 2017). The simplest method of question classification is to use a set of rules that map patterns of questions into question types by analyzing the interrogative terms of the question (wh-terms). However, developing such rules takes time, and adapting to a new domain is challenging. An alternative approach to question classification is the use of machine-learning techniques. This approach treats question type identification using statistical classification packages like a support vector machine (Abedissa and Libsie, 2019). Finally, the question analysis component generates queries from the given question by selecting keywords and removing interrogative terms. In addition, expand the set of keywords using synonyms (Utomo et al., 2017).

The document retrieval component is a standard document retrieval system that identifies a subset of documents that contain terms of a given query from the total document collection deemed most likely to have an answer to the question (Utomo et al., 2017). While trying to identify relevant information more accurately, it splits the documents into several passages and treats the passages as documents. Using a passage-based retrieval approach instead of a full-document retrieval approach has the advantage of returning short text excerpts instead of entire documents, which are easier to process by later components of the question-answering system (Chen et al., 2017).

The answer processing component takes retrieved documents likely to contain an answer to the question and specifies what types of phrases should count as correct answers. Then, it extracts several candidate answers, ranks them in their probable correctness, and returns an answer from those top-ranked phrases. Answer extraction and selection are treated as a classification or ranking problem and solved using heuristics and machine learning methods. Since deep neural networks learn to select features by end-to-end training, most recent QA models use a neural architecture to encode contexts and questions into a vector space and reason over them (Mozannar et al., 2019).

In the era of deep learning, especially pre-trained language models like BERT (Kenton and Toutanova, 2019) enables robust QA model development (Laskar et al., 2020). Besides, the introduction of multi-lingual language models like Cross Language Multilingual-Roberta (XLM-R) (Conneau et al., 2020) and mBERT (Wu and Dredze, 2020) contribute to the advancements of the cross and multi-lingual QA. Existing deep learning-based QA systems fall into retriever-reader, dense retriever and end-to-end training, and retriever-free approaches (Chen and Yih, 2020).

The retriever-reader-based QA models first retrieve relevant passages, then read top-ranked passages and predict the beginning and end positions of the answer text from a context. DrQA (Chen et al., 2017) is a typical example of this approach. In the DrQA model, the retriever uses traditional sparse vector space methods, representing every question and document as bag-of-words vectors weighted by TF-IDF (term frequency-invert document frequency). Then, the retriever passes five top-ranked documents to the reader component. The reader uses a 3-layer bidirectional long-short-term memory (LSTM) (), which encodes the question and the top-ranked paragraphs as a sequence of feature vectors. Then, it predicts the probability of the start and end positions of the answer span (Cui et al., 2019).

The QA models in the retriever-generator approach follow the major paradigm shift towards neural-based IR. To answer a question, generate the response using a retrieved-context instead of predicting start/end positions (Lewis et al., 2020b).

Unlike the retriever-reader and retriever-generator approaches, the generative approach generates free text as an answer to respond to questions using the knowledge in its parameters (Roberts et al., 2020). To test the capability of memorizing factual knowledge of pre-trained language models, Roberts (Roberts et al., 2020)

fine-tuned the T5 (Text-to-Text Transfer Transformer) (Raffel et al., 2020) language model to answer questions without providing it with any additional information or context.

Specific to Amharic, there are very few QA models (Abedissa and Libsie, 2019; Elema, 2022; Yimam and Libsie, 2009); however, none provide a public dataset. Therefore, this paper introduces the first factoid extractive open-domain **Amh**aric **Qu**estion **A**nswering **D**ataset (Amh-QuAD), the dataset can be found online at https://github.com/semantic-systems/amharic-qa.

As shown in Figure Figure 1, the Amh-QuAD dataset comprises context, question, and answer triplets. The contexts consist of articles gathered from Amharic Wikipedia[1], while we crowdsource 2628 question-answer pairs from 378 contexts. For example, for the question given in Figure Figure 1, "ከላሊበላ አስራ አንድ ውቅር አብያተ ክርስቲያናት የመስቀል ቅርጽ ያለው የትኛው ነው?" (Of the 11 Lalibela's rock-hewn churches, which one is cross-shaped?), the answer "ቤተ ጊዮርጊስ" (betə giorgis 'House of St. George') is the span from the context. In our work, in addition to the crowdsourced question-answer pairs, we have set baseline F1-score values by implementing a QA model with the retriever and reader components. We fine-tuned the XLMR model for the reader component using the Amh-QuAD training set and achieved an 81.34 F-score value.

2. Amharic Interrogative Sentences

Amharic, an indigenous African language from Ethiopia, has its unique writing system using the Ge'ez script known as ፊደል (Fidel). As shown in Figure Figure 2, an Amharic interrogative sentence is formulated using information-seeking pronouns like "ምን" (what), "መቼ" (when), "ማን" (who), "የት" (where), "የትኛው" (which) etc. or prepositional interrogative phrases like "ለምን" [ለ-ምን] (why), "በምን" [በ-ምን] (by what), etc. Also, verb phrases such as ግለጽ (explain), ዘርዝር|ሪ (list), አንፃፅር|ሪ|ሩ (compare), etc. are used to pose questions (Yimam, 2009; Amare, 2013).

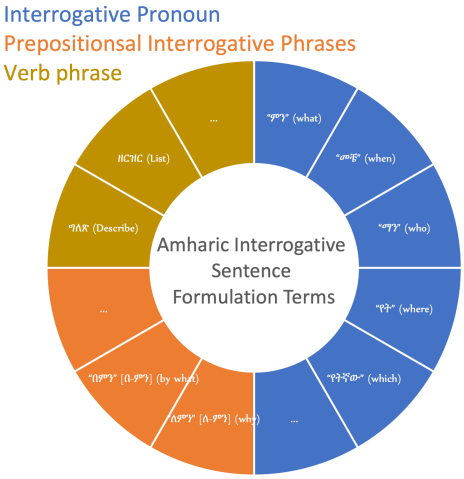

Figure 2: Amharic Interrogative Terms.

3. Related Work

Among the existing English QA datasets, SQuAD (Rajpurkar et al., 2016) paved the way by creating question-answer pairs from Wikipedia articles using crowd workers, where each question answer is a span of text in the articles. Chinese MRC (Cui et al., 2019), Vietnamese QA (Do et al., 2021), and other data sets listed in (Dzendzik et al., 2021; Rogers et al., 2023) also follow the same curation step as SQuAD. Following crowdsourcing, TigQuAD (Gaim et al., 2023) introduces a QA dataset for the low-resourced Semitic language Tigrinya from newspapers. Amharic and Tigrinya are both Semitic languages. However, the linguistic differences in the writing scripts of the two languages (Feleke, 2017) hinder TigQuAD from being used for testing and training Amharic QA models.

On the other hand, by automatically translating SQuAD into their respective languages, German (Möller et al., 2021) and French (d'Hoffschmidt et al., 2020) versions have been created. The Arabic QA dataset (Mozannar et al., 2019) is created partly by translating from SQuAD and partly by crowdsourcing. Translating existing QA datasets to other languages is one solution for creating a large data set. However, we opt for the crowdsourcing approach due to the absence of a well-tested open-source English-to-Amharic machine translation tool.

In Amharic, there are very few QA models; TETEYEQ (Yimam and Libsie, 2009) answers factoid-type questions by extracting entity names using a rule-based answer extraction approach. Abedissa and Libsie(2019) introduce a non-factoid QA model that answers biography, description, and definition questions. The definition-description answer extraction uses heuristics; meanwhile, it answers biography questions using a summarizer and validates the summary with a classifier. The work in (Elema, 2022) classifies questions using a neural network model, selects candidate answers by a hybrid Bi-LSTM and CNN model, and extracts answers as named entities utilizing a named entity recognizer. Unlike the existing Amharic QA systems, this study proposes a retriever-reader-based Amharic QA (AmhQA) that leverages a multi-lingual language model (Conneau et al., 2020). Beyond

[1] https://am.wikipedia.org/

attempting to answer Amharic questions, work has yet to produce a published dataset suitable for training and testing the performance of Amharic QA models. Therefore, we present Amh-QuAD as a train & test benchmark for Amharic QA models.

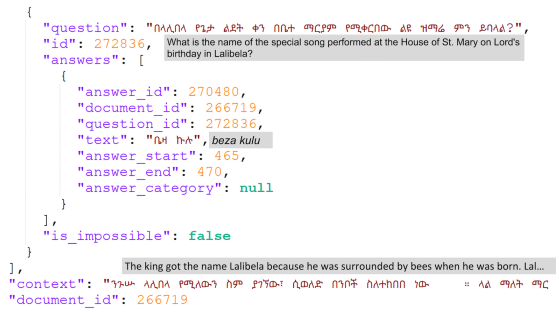

Figure 3: The Amh-QuAD structure.

4. The Amh-QuAD

The Amh-QuAD dataset is created in three phases: article gathering, crowdsourcing question-answer pairs, and annotation.

4.1. Collection and Cleaning

We collect the Amharic articles used as contexts from the Amharic Wikipedia dump[2]. We keep only those articles larger than 2 KB and whose category is not “proverb” and “food preparation”. Proverb articles are advantageous for generating reasoning questions. Additionally, ‘food preparation’ articles mainly consist of steps for preparing food, making them suitable for generating questions such as ‘How is the step to cook...’ and ‘List the steps or ingredients added while cooking...’. Also, in both scenarios, the answer is not confined to a continuous text span within the article but instead spreads out among non-consecutive sentences. We further preprocess the remaining articles after filtration using the wiki-dump-reader tool[3] to obtain clean texts. Subsequently, as long articles do not comprehensively stimulate question creation, each article is segmented based on its sub-topics. Finally, we randomly selected 378 cleaned articles.

4.2. Question-Answer Pair Crowdsourcing

We provide training on formulating questions that can be answered by a given context, following the Haystack guideline[4]. Since we randomly select articles from Wikipedia, we inform annotators to flag any articles containing offensive content. Additionally, we encourage annotators to generate as many questions as possible from a given context.

4.3. Question-Answer Pair Validation and Annotation

The validation of the formulated question-answer pairs is about their correctness. We say a question is correct if the posed question is answerable by the given context, grammatically correct, and clearly defines the subject or object under consideration. For example, a question like ‘How many parks does our (the) country have?’ is ambiguous due to the possessive adjective ‘our’ or the definite article ‘the’; it is challenging to know to which country it refers. We paraphrase such questions according to the context, besides rewriting the questions that do not explicitly state the subject or object. In addition, we exclude questions that are too long and have non-consecutive string answers from the annotation. Then, annotate the question-answer pairs using the Haystack annotation tool[5]. As shown in Figure 3 The annotation tool provides the annotated question-answer pairs as JSON files in SQuAD format.

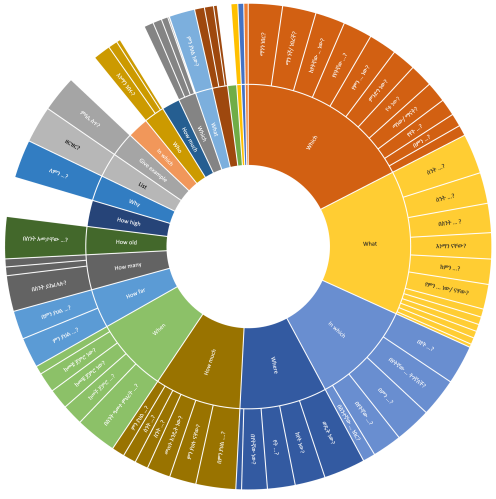

Figure 4: Interrogative terms distribution.

4.4. Data Analysis

As shown in Table Table 1, the Amh-QuAD contains 378 articles and 2628 question-answer pairs. The contexts, on average, have 172 words. Most questions’ average word length is 9.22, whereas the

[2] https://dumps.wikimedia.org/amwiki/20210801/ last accessed 18 August 2021

[3] https://pypi.org/project/wiki-dump-reader/

[4] https://drive.google.com/file/d/1Wv3OIC0Z7ibHIzOm9Xw_r0gjTFmpl-33/view

[5] https://docs.haystack.deepset.ai/docs/annotation

	Article	Question	Answer
Size	378	2628	2628
Word len (avg)	172.07	9.22	2.66

Table 1: Size and average word length of articles, questions, and answers.

answers are short, and their average word length is 2.66. Furthermore, we split the dataset into train, dev, and test with a size of 1728, 600, and 300, respectively. Besides, to see the distribution of interrogative terms on the test set, we manually identify the interrogative term and examine the adjacent two or three tokens. Figure Figure 4 illustrates the distribution of the interrogative terms, showcasing the variety in the question terms within Amh-QuAD.

4.5. Questions Expected Answer Type

Examining the question's interrogative terms and answers, we categorize the 300 test questions into person, location, time, organization, number, description, and other classes. Then, compute the percentage of the coverage of the expected answer types in the test set. As shown in Figure Figure 5, we found that most questions are about Location, Number, and Time, where each type has over 18% coverage. Description questions take 13% of the share and questions that seek a person's name as an answer are 14%. 10% of questions like "What is the working language of Ethiopia?" whose expected answer types are entities that cannot be included in the existing categories and fall into the 'OTHER' group. The list (3%) and organization (3%) are the smallest among the questions.

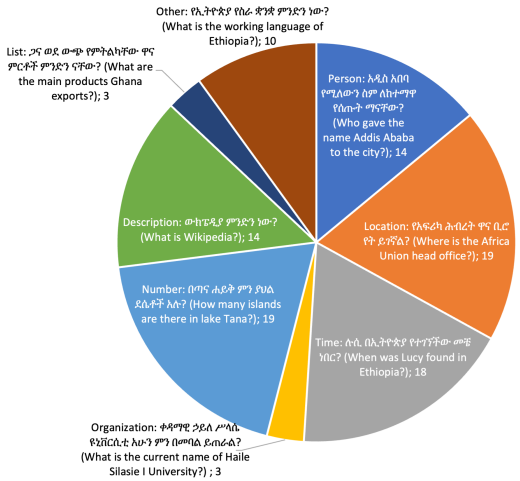

Figure 5: Question Types Distribution in %.

5. Amharic QA Model

Problem Definition: Given a question Q and a set of contexts C, the goal of the Amharic QA (AmhQA) model is to retrieve top-k relevant C_i from C and predict a span of text from the retrieved C_i's that answers Q.

1. Retrieve top-k relevant contexts using a retriever:

$$C_i \in C \quad \text{where} \quad i = 1, 2, \ldots, n$$

2. Predict a span of text from the retrieved context via a reader:

$$S_i = \text{predict}(Q, C_i)$$

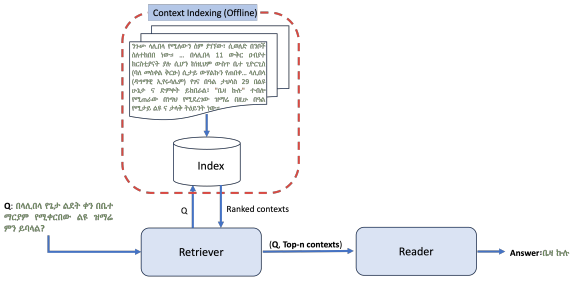

Figure 6: The Amharic QA model.

As shown in Figure 6, the AmhQA model has three components: offline indexer, retriever, and reader.

5.1. Indexing

Offline indexing begins by obtaining contexts from the test dataset. Then, an NLTK-based preprocessor tokenizes the contexts at the word level and splits the contexts into smaller segments based on a maximum length of 200 words with no word overlap. Finally, index the segmented contexts using the Elasticsearch Indexer[6]. This indexing process creates an inverted index, enabling rapid and efficient retrieval of relevant information during subsequent queries.

5.2. Retriever

The AmhQA retriever component utilizes the BM25 (Best Match 25)algorithm to return the top-k most relevant contexts. The BM25 algorithm is a modified TF-IDF(Term Frequency - Inverse Document Frequency) that scores and orders contexts based on their relevance to the given question (Robertson and Zaragoza, 2009). The retriever calculates a

[6] https://www.elastic.co/blog/what-is-an-elasticsearch-index

relevance score for each document by considering term frequencies within documents, document length normalization, and term saturation. Term saturation is the concept that a term's relevance to a document decreases as it appears more frequently within the document. The term saturation function modifies the term frequency during the relevance score calculation.

For a question Q and context C_i, the BM25 scoring formula is:

$$\text{score}(Q, C_i) = \sum_{i=1}^{n} \text{idf}(q_i) \cdot \frac{f(q_i, C_i) \cdot (k_1 + 1)}{f(q_i, C_i) + k_1 \cdot \left(1 - b + b \cdot \frac{|C_i|}{\text{avgcl}}\right)}$$

Where:

- n is the number of terms in the question and q_i is the i-th term in the question.
- $\text{idf}(q_i)$ is the inverse document frequency of term q_i.
- $f(q_i, C_i)$ is the term frequency of term q_i in context C_i.
- $|C_i|$ is the length of context C_i.
- *avgcl* is the average length of contexts in the collection; k_1 and b are tuning parameters.

The parameters k_1 and b control the term frequency component of the scoring. k_1 is a positive tuning parameter that regulates the saturation effect of term frequency. A higher value of k_1 increases the impact of term frequency on the scoring, making the algorithm more sensitive to the frequency of terms in the document. Conversely, a lower value of k_1 reduces the impact of term frequency, leading to less effect on the scoring. The parameter b is a value between 0 and 1 that controls the influence of document length normalization. When b is closer to 0, document length normalization has a weaker effect, resulting in less attenuation of the term frequency component for longer documents. On the other hand, when b is closer to 1, document length normalization has a more substantial effect, causing the term frequency component to favor longer documents.

The retrieved documents are then ranked based on the value of *score(Q, C_i)*.

5.3. Reader

The AmhQA reader component is created by fine-tuning an instance of the XLM-R pre-trained language model from Hugging Face[7] using the open source Haystack framework[8] on our training set. The XLM-R (Cross-lingual Language Model - RoBERTa) (Conneau et al., 2020) is a transformer-based language model trained on diverse languages, including Amharic. While fine-tuning, we use the default settings of the Haystack framework. The reader model generalizes for unseen examples despite being fine-tuned on a small dataset comprising 1728 samples. During the answer retrieval, the reader tokenizes the question Q and context C_i, encodes the tokenized question and context, and produces probability distributions of the answer span start and end indices. Finally, it decodes the answer span indices into human-readable text based on the highest probability span.

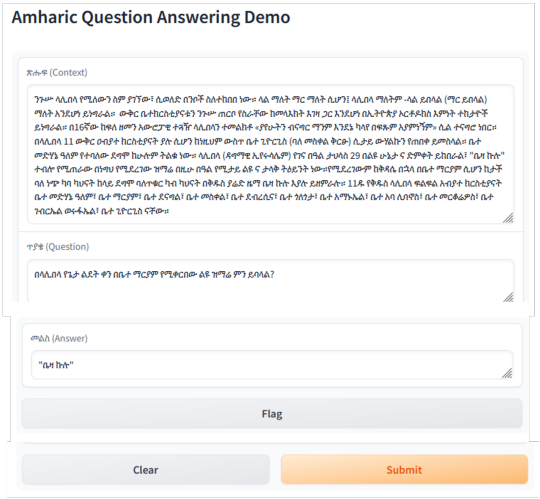

Figure 7: AmhQA Prototype Interface.

6. Experiment

6.1. Baseline Model

Since the Amh-QuAD dataset contains a set of contexts and question-answer pairs, its inherent task is reading comprehension (RC) (Rajpurkar et al., 2016). That is, given a question Q and a context C, the goal of the model is to identify a word or group of consecutive words from C that answers Q. Hence, based on this assumption, we have set a baseline value for the Amh-QuAD using an XLM-R-based RC and with our fine-tuned reader model[9]. Figure 7 shows the RC setting of the AmhQA model prototype interface.

On the other hand, our retriever-reader (RR) based AmhQA model first retrieves relevant passages and then reads top-ranked passages to predict the start and end positions of the answer. The retriever part is based on BM25, and the reader is implemented using our fine-tuned reader model.

[7] https://huggingface.co/deepset/xlm-roberta-large-squad2

[8] https://github.com/deepset-ai/haystack/

[9] https://huggingface.co/deepset/xlm-roberta-large-squad2

Settings	EM	F1
XLM-R on MLQA	52.70	70.7
RC (XLM-R$_{Base}$)	47.49	64.69
RC (XLM-R$_{Large}$)	56.52	74.35
RC (With Fine-tuned Reader)	**67.89**	**81.34**
RR (With Fine-tuned Reader)	67.4	80.3

Table 2: AhmQA performance in RC and RR settings.

6.2. Evaluation and Discussion

The goal of evaluating a QA model is to measure the model's accuracy and its components. For QA datasets where the answer is a span of a text, an exact match (EM) with the gold answer is widely utilized (Rajpurkar et al., 2016). The EM metrics have an all-or-nothing drawback. To overcome this, precision, recall, and their harmonic mean, the F-Score value, is also used (Chen et al., 2017). Recall (R) gives the fraction of words that the system has chosen from the totality of words found in the actual answer, and precision (P) measures the fraction of system answers that are correctly chosen. Besides, Mean Reciprocal Recall (MRR) and Mean Average Precision (MAP) metrics evaluate the retriever performance.

As shown in Table 2, on the RC setting the XLM-R$_{Large}$ F1 score is 74.35, whereas the XLM-R$_{Base}$ F1 score is 64.69. The F1 score of the XLM-R$_{Large}$ on the Amh-QuAD test set was comparable to its average F1 score (70.7) on the MLQA dataset for other seven languages (Lewis et al., 2020a). Our fine-tuned reader also led to substantial improvements, yielding an EM score of 67.89 and an F1 score of 81.34. Even though the difference in the F1 scores achieved by the RC (81.34) and the RR (80.3) settings is minimal, one reason is the segmentation of contexts without overlap during indexing in the RR configuration. The segmentation can split the answer strings into non-overlapping segments, making it difficult for the RR to extract accurate answers. Unlike that, the RC model uses whole context embedding to extract answers from passages, enabling it to achieve better results. Furthermore, the RR includes the retrieval and reading components, introducing complexities in integrating and processing retrieved contexts that affect performance.

6.3. Ablation Study

As shown in Table 3, when the retriever number of context retrieval configuration is top-1, MRR and MAP are high at 82.9, indicating their effectiveness in correctly ranking and retrieving relevant information. Moreover, when expanding the retrieval to the top three results, the scores increase even further. The MRR and MAP reach 88.4 and 88.2, respectively, which indicates that considering multiple retrieval options improves the retriever's ability to capture a broader range of relevant documents, resulting in better ranking and precision. The significant improvement in performance from the top-1 to top-3 settings highlights the necessity of considering multiple retrieval options to optimize the effectiveness of the retriever in the QA models.

	MRR	MAP
top-1	82.9	82.9
top-3	88.4	88.2
top-3**	88.4	88.2

Table 3: AhmQA Retriever component ablation. **(With Fine-tuned Reader)

	EM (top-1)	F1 (top-1)	EM (top-3)	F1 (top-3)
top-1	48.0	60.7	-	-
top-3	53.0	66.6	58.72	73.22
top-3**	50.7	60.9	**67.4**	**80.3**

Table 4: AhmQA Reader component ablation. **(With Fine-tuned Reader)

Table 4 shows the reader component's performance across various metrics and retrieval settings. When considering only the top-1 retrieved context, the Exact Match (EM) and F1 scores are 48.0 and 60.7, respectively. Expanding the retrieved context to the top three results increases the EM and F1 scores at top-1 to 53.0 and 66.6, respectively. Furthermore, when evaluating based on the top three retrieved contexts, both EM and F1 scores experience significant improvements, reaching 58.72 and 73.22, respectively. Highlights the importance of considering multiple retrieved contexts for optimizing the reader's performance, as it allows for a more comprehensive synthesis of contexts.

The fine-tuned reader component has demonstrated a significant performance improvement compared to the previous evaluation. Specifically, the exact match (EM) score has increased to 67.4, indicating higher accuracy in providing precise answers. The F1 score has also improved, reaching 80.3, reflecting enhanced effectiveness in generating correct answers. The top-1 evaluation metrics also show improvements, with exact match top-1 and F1 top-1 scores increasing to 50.7 and 60.9, respectively. These results emphasize the enhanced performance of the fine-tuned reader across different evaluation settings. Overall, the results showcase the improvements achieved through fine-tuning, indicating a more reliable reader component for Amharic QA.

7. Summary

The Amh-QuAD dataset is an effort towards inclusiveness and accessibility in natural language processing (NLP). The development of this dataset will partly address the imbalance in language resources, particularly for underrepresented languages within the NLP community. The Amh-QuAD is the first publicly available factoid open-domain extractive Amharic QA dataset containing triplets of context, question, and answer curated from Amharic Wikipedia, which serves in RC and retriever-reader QA settings. In addition, we introduce a new AmhQA reader by fine-tuning a multilingual pre-trained language model. Also, set baseline values in reading comprehension and retriever-reader QA settings.

8. Bibliographical References

Tilahun Abedissa and Mulugeta Libsie. 2019. Amharic Question Answering for Biography, Definition, and Description Questions. In Fisseha Mekuria, Ethiopia Nigussie, and Tesfa Tegegne, editors, *Information and Communication Technology for Development for Africa*, volume 1026, pages 301–310. Springer International Publishing, Cham. Series Title: Communications in Computer and Information Science.

Getahun Amare. 2013. *Amharic Grammar in Simple Way*. Addis Ababa, Addis Ababa University Press.

B Barla Cambazoglu, Mark Sanderson, Falk Scholer, and Bruce Croft. 2020. A Review of Public Datasets in Question Answering Research. *ACM SIGIR Forum*, 54(2):23.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Danqi Chen and Wen-tau Yih. 2020. Open-Domain Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 34–37, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. "a span-extraction dataset for Chinese machine reading comprehension". In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5883–5889, Hong Kong, China. Association for Computational Linguistics.

Martin d'Hoffschmidt, Wacim Belblidia, Quentin Heinrich, Tom Brendlé, and Maxime Vidal. 2020. FQuAD: French question answering dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1193–1208, Online. Association for Computational Linguistics.

Phong Nguyen-Thuan Do, Nhat Duy Nguyen, Tin Van Huynh, Kiet Van Nguyen, Anh Gia-Tuan Nguyen, and Ngan Luu-Thuy Nguyen. 2021. Sentence Extraction-Based Machine Reading Comprehension for Vietnamese. In *Knowledge Science, Engineering and Management: 14th International Conference, KSEM 2021, Tokyo, Japan, August 14–16, 2021, Proceedings, Part II*, page 511–523, Berlin, Heidelberg. Springer-Verlag.

Daria Dzendzik, Jennifer Foster, and Carl Vogel. 2021. English Machine Reading Comprehension Datasets: A Survey. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8784–8804, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Abenezer Mengistu Elema. 2022. Developing Amharic Question Answering Model Over Unstructured Data Source Using Deep Learning Approach. In *2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, pages 108–113.

Alexander Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-Based Question Generation from Retrieved Sentences for Improved Unsupervised Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4508–4513, Online. Association for Computational Linguistics.

Tekabe Legesse Feleke. 2017. The similarity and mutual intelligibility between Amharic and Tigrigna varieties. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 47–54, Valencia, Spain. Association for Computational Linguistics.

Fitsum Gaim, Wonsuk Yang, Hancheol Park, and Jong Park. 2023. Question-answering in a low-resourced language: Benchmark dataset and models for Tigrinya. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11857–11870, Toronto, Canada. Association for Computational Linguistics.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2.

Md Tahmid Rahman Laskar, Jimmy Xiangji Huang, and Enamul Hoque. 2020. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5505–5514, Marseille, France. European Language Resources Association.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020a. MLQA: Evaluating Cross-lingual Extractive Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Timo Möller, Julian Risch, and Malte Pietsch. 2021. GermanQuAD and GermanDPR: Improving non-English question answering and passage retrieval. In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, pages 42–50, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hussein Mozannar, Elie Maamary, Karl El Hajal, and Hazem Hajj. 2019. Neural Arabic question answering. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 108–118, Florence, Italy. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv:1606.05250 [cs]*. ArXiv: 1606.05250.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. QA Dataset Explosion: A Taxonomy of NLP Resources for Question Answering and Reading Comprehension. *ACM Comput. Surv.*, 55(10).

Fandy Setyo Utomo, Nanna Suryana, and Mohd Sanusi Aami. 2017. Question Answering System: A Review on Question Analysis, Document Processing, and Answer Extraction Techniques. *Journal of Theoretical & Applied Information Technology*, 95(14).

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Baye Yimam. 2009. *Amharic Grammar*. Addis Ababa, Addis Ababa University Press.

Seid Muhie Yimam and Mulugeta Libsie. 2009. TETEYEQ: Amharic question answering for factoid questions. *IE-IR-LRL*, 3(4):17.