Exploring Open-Domain Question Answering in Hebrew

Novel Dataset Explored with AlephBERT

Ofri Masad (ID. 036102192), Gal Rapoport (ID. 304786353)

Submitted as final project report for the NLP course, IDC, 2021

1 Introduction

In this paper we propose a new approach for solving the **Open Domain Question Answering** task for the Hebrew language. Although Hebrew language models[6] were already introduced a few years ago, the lack of Hebrew data resources for this task and the relatively low amount of research done on Hebrew language models, produced no available solution for the Question Answering task in Hebrew (as far as we could find).
We propose a retriever-reader system, based on the Hebrew Wikipedia, indexed by the Anserini [9] infrastructure, coupled with a HuggingFace QuestionAnswering model based on the pretrained AlephBert (Hebrew language model) [6].
We focused our efforts on the lack of dataset, trying to produce a question-answering dataset large enough and high quality enough to train a language model. We have experimented with a few different datasets, which were machine translated and preprocessed. In our search for the best possible translation settings, we have turned to an Arabic language dataset and translated it into Hebrew. Our assumption, which proved itself to be correct, was that the Arabic language, being much closer to Hebrew in its linguistic characteristics, will produce better results.

1.1 Related Works

Assuming our model is not big enough nor expressive enough to actually "memorize" the entire knowledge base required in order to answer questions, we need to use the Retriever-Reader approach. This concept is also referred to as "Open-Book QA". Like in an open-book test, the "student" can look for the answer in the book (i.e., context) and not only in his head.

One famous implementation of that concept is DrQA [1] (code). Chen et al. have used an efficient non-learning vector space model. Every query and document is modelled as a bag-of-words vector, and TF-IDF is used in order to find the most suitable document to be used as a context for the query.

The later BERTSerini [7] (Yang et al., 2019) pairs the open-source Anserini toolkit [8] as the retriever with BERT[3] type model. The retrieved text segments are ranked by BM25, a classic TF-IDF-based retrieval scoring function. This approach produced better exact-match (EM) results than the former DrQA[1], and is more suitable for smaller datasets (such as the Hebrew dataset).

We have used two datasets:

1. SQuAD v2.0 dataset [5] containing about 130K questions, many of which are labeled as impossible (i.e., usually the answer is not in the context). This dataset is commonly used for the task and contain only English text.

2. TyDi QA dataset [2], containing questions and answers in 11 languages, one of which is Arabic, which we used for translation. The Arabic train set contains only 14K questions and answers.

2 Solution

2.1 General approach

We have followed the work on BERTSerini [7] and designed a system where the retriever is the Anserini toolkit [8] with an indexed Hebrew Wikipedia, and the reader is a pretrained AlephBERT model [6]. The Wikipedia text had to be stripped from links, special characters and Hebrew Niqqud. We train the model on the two translated dataset which were preprocessed for the task.

The SQuAD dataset [5] is usually the first benchmark when it comes to QA tasks. We have translated each context, question and answer with several free online tools. After a manual assessment of the results, we decided using one of Google Translate API's versions. The results were stripped from Hebrew Niqqud, and the index of the answer in the context was updated to fit the Hebrew version. Moreover, some questions had to be removed since the answers were no longer available in the context after translation.

Entity Translation:
One common issue with the automated translation was with entity translation. More specifically, the short questions and answers were falsely translated. Our favorite example was:

English: *"when did the rams won the super bowl".*
Hebrew: "מתי זכו האיילים בקערת העל". ‘

To avoid this phenomena, we concatenate the context together with the questions and answers (related to that context), translate the full text and then split the translated result back to pieces. After that process the same sentence is translated as follows:

Hebrew: "מתי זכו הראמס בסופרבול"

Tokenization:
Another known issue with Hebrew NLP tasks is the tokenization. The Hebrew Tokenizer by AlephBERT[6] is statistical and not linguistic and thus it underperforms. In addition it has 52K tokens, comparing with standard BERT[3] 30.5K word pieces. AlephBERT creators have stressed this issue and left it as a required improvement for future work.

Transitivity from Arabic:
Although the machine translated data was in most cases human-readable, it was very poorly constructed as a Hebrew text. Consequently, it produced results which were significantly less accurate than the English version of the dataset.

Our hypothesis is that the very different nature of the English and Hebrew languages was making the translation less natural and comprehensible.
Although machine translation (MT) models have made an amazing progress within the past few years, bridging the gap between Hebrew and English may take some time. The Hebrew language has a very different structure than the Latin languages. Hebrew word order is more flexible than the rigid syntax of the English language. Adjectives usually comes after nouns. Hebrew has more verb postfixes, nouns and pronouns vary in postfix according to the preposition that precedes them, and adjectives must agree in plural/singe and gender with the nouns they modify.

Therefore, we explore the notion of translating from a much closer language to Hebrew - from the Arabic language. The Arabic language is more common than the Hebrew language (i.e., 420M Arabic speakers and 9M Hebrew speakers). Moreover, as some native Arabic speaking countries are considered as developing countries - it is much easier and cost-effective to crowd-source the generation of a new dataset in Arabic. Perhaps this is one of the reasons why HuggingFace showcase about 60 Arabic datasets, while only 23 Hebrew datasets, most of which are machine translated, are available. Nevertheless, it is clear that more efforts are invested in developing English-to-Hebrew translators and so this gap, if still exist, will be closed in the upcoming future.

Hebrew and Arabic are both Semitic languages. Both Hebrew and Arabic rely on three-letter roots. This means you would find the same letters appearing in words with related meaning. Hebrew and Arabic share some letters of the alphabet - in some cases, letters which are not available in English, like the Hebrew Tsade and the Arabic Saad. The general structure of sentences is closer between these two languages, and the usage of prefixes and postfixes for gender, plural, preposition and more is very similar in many cases.

We have translated 14K questions and answers from the Arabic part of the TyDi-QA dataset [2] into Hebrew and trained our model with this new dataset. The achieved results were much better.

We provide plots and tables detailing our results in the following section. Nevertheless, it is important to stress that we could not find a dataset which is available both in English and in Arabic (both in the original, non translated form). Even datasets which contain both English and Arabic, have different questions in each language. In addition, there is a different amount of questions in each language. Thus, it is hard to accurately compare the results achieved by translating from these two languages and training on the translated data.

2.2 Design

We have used the pretrained AlpehBERT[6] model with hidden size of 768,
12 hidden layers and 12 attention heads (as in BERT[3]). The vocabulary used by AlephBERT is of size 52k (much larger than BERT's 30.5k tokens). We replaced the BERT model[3] inside BertSerini[7] with our model and had to change the token-type embedding input, since AlpehBERT did not use next sentence prediction losses and thus did not use the *SEP* token. This loss was used in the original BERT's training but was shown to have a minor affect on the results in the work of Liu et al. on RoBERTa[4].

The tokenized question is concatenated with a sub-sequence from the context which holds the answer, separated by a special token.
This process id done by the SQuAD python package. The loss is calculated (again, by SQuAD[5]) according to the position (index) of the start end end token describing the answer location within the context sub-sequence.

To index the Hebrew Wikipedia we used *pyserini*[9], a python wrapper around the Anserini Java package[8]. The Wikipedia text had to be stripped from links, special characters and Hebrew Niqqud. To use the model in interactive mode, we first search for the question using the index. We extract the top-10 Wikipedia articles and try to use them as the context for prediction by their order of retrieval. We used the first non-empty answer from our model's outputs (notice: the question answering pipeline by HuggingFace may output an empty answer).

3 Experimental results

We conducted numerous training and testing sessions with different hyper parameters. The typical training time on a NVIDIA GeForce RTX 2080Ti was around 2 hours (3 SQuAD epochs or 18 TyDi-QA epochs), but we observed overfitting after 2 epochs of each dataset. Since TyDi-QA is very small dataset, the overfitting starts very early in the training process. We tried freezing most layers of the model but the yielded result were much lower. Other regularization methods such as dropout, or setting a higher weight decay did produce lesser results or were inapplicable as well.

Data split		
Dataset	Train	Dev
SQuAD-He	54086 (18352)	5610 (338)
TyDi-QA-He	7861	882

Table 1: **Train - Dev Dataset Split**
SQuAD contains some questions which are defined as ”impossible” (counted in parentheses). ”Impossible” questions are approximately 33% of train set.

Training result					
Train set	trans. from	Test set	trans. from	F1	EM
SQuAD-train	-	SQuAD-dev	-	44.16	40.21
SQuAD-He-train	En	SQuAD-He-dev	En	27.71	21.21
TyDi-QA-He-train	Ar	SQuAD-He-dev	En	13.69	9.6
TyDi-QA-He-train	Ar	TyDi-QA-He-dev	Ar	42.25	34.06
SQuAD-He-train	En	TyDi-QA-He-dev	Ar	33.22	23.96

Table 2: **Training Results**
The first row is the baseline trained on an English dataset. The other rows present the results on SQuAD dataset translated from English, and TyDi-QA dataset translated from Arabic.

We used the common metrics to measure our results. Every prediction is scored based on exact match(EM) and token overlap (F1 score).
The results are presented in Figure 1 and Table 2.

We crossed tested the training and dev sets in order to get a better understanding of the affect each one had on the results.

The results shown in this section were achieved with lr=3e-5, batch-size=12 and weight-decay=0

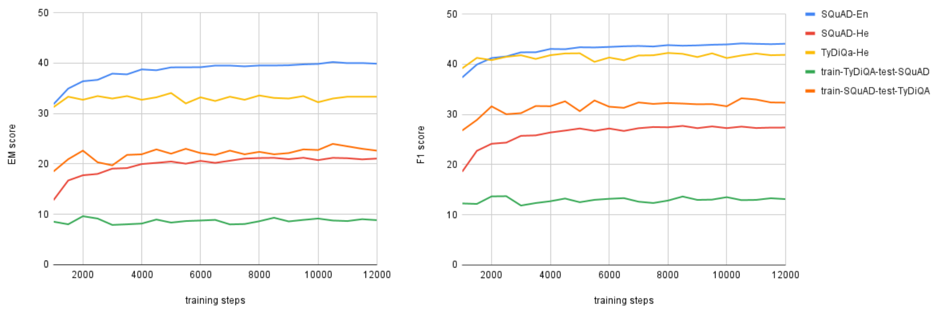


Figure 1: EM [left] and F1[right] values during training

4 Discussion

In this work we described and evaluated a new approach for Open-Domain Question-Answering task in Hebrew.

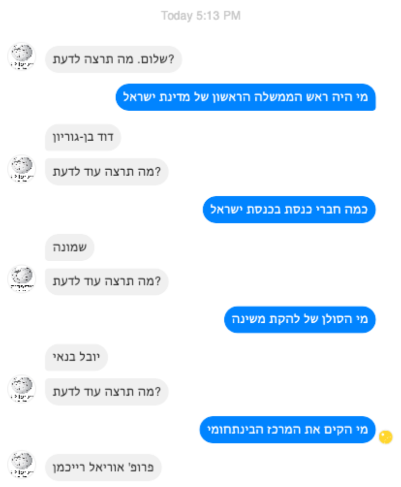
We suggested translating datasets from the English and Arabic languages and train on those machine translated datasets. From our experiments, it seems that the model was able to train and properly generalize on the translated datasets. Although our results did not achieve the same accuracy as of the training on the original datasets, when considering the inherent problems in Hebrew NLP and the fact that the data was machine translated, the results seems comparably good.

It seems that the TyDi-QA[2] dataset is easier than SQuAD[5] dataset (per training, and achieving satisfying results). This may be the result of the machine translation process, but it is more likely to be due to the nature of the original dataset.

Between the two session which where tested on SQuAD-He-dev set, the training on TyDi-QA had produced much inferior results. This may be due to the size of the dataset, the level of the original dataset or the difference in the translation origin. When testing on TyDi-QA-He-dev the results were better than with SQuAD-He-dev results. This supports the hypothesis that TyDi-QA is naturally easier than SQuAD (in the origin language). Yet, when comparing between the two test sets on TyDi-QA-He-dev, it seems that the model which was trained on TyDi-QA-He-train had produced significantly better results. This supports the hypothesis that the translation of TyDi-QA is better than the translation of SQuAD, or at least, very different than the translations of SQuAD.

4.1 Demonstration

Our AlephBERTSerini is able to answer simple questions relatively successfully.



Today 5:13 PM

שלום. מה תרצה לדעת?

מי היה ראש הממשלה הראשון של מדינת ישראל

דוד בן-גוריון

מה תרצה עוד לדעת?

כמה חברי כנסת בכנסת ישראל

שמונה

מה תרצה עוד לדעת?

מי הסולן של להקת משינה

יובל בנאי

מה תרצה עוד לדעת?

מי הקים את המרכז הבינתחומי

פרופ' אוריאל רייכמן

Figure 2: Chat with Hebrew AlephBERTSerini.

5 Code

Our code is provided in the following link:
https://github.com/ofrimasad/hebrew_qa.

References

[1] Chen et al. “Document Retriever Question Answering.” In: (2017). DOI: https://arxiv.org/abs/1704.00051.

[2] Clark et al. “TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages.” In: (2020). DOI: https://arxiv.org/abs/2003.05002.

[3] Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: (2018). DOI: https://arxiv.org/abs/1810.04805.

[4] Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” In: (2019). DOI: https://arxiv.org/abs/1907.11692.

[5] Rajpurkar et al. “SQuAD: 100,000+ Questions for Machine Comprehension of Text.” In: (2016). DOI: https://arxiv.org/pdf/2104.04052.

[6] Seker et al. “AlephBERT: A Hebrew Large Pre-Trained Language Model to Start-off your Hebrew NLP Application With.” In: (2021). DOI: https://arxiv.org/pdf/2104.04052.

[7] Yang et al. “End-to-End Open-Domain Question Answering with BERTserini.” In: (2019). DOI: https://arxiv.org/abs/1902.01718.

[8] Castorini. *AnSerini*. URL: https://github.com/castorini/anserini.

[9] Castorini. *PySerini*. URL: https://github.com/castorini/pyserini/.