Unsupervised Question Answering Using Custom NLP Library Built for Egyptian Arabic

Stanford CS224N Custom Project

Ahmed Sharaf
Department of Computer Science
Stanford University
ammas@stanford.edu

Michael Souliman
Department of Computer Science
Stanford University
msoul@stanford.edu

Abstract

Egyptian Arabic is the most spoken dialect of Arabic in the Middle East with over 110 million speakers. However, unlike Modern Standard Arabic (MSA), there are no NLP tools or applications for Egyptian Arabic. This is due to the fact that there are significant differences between MSA and Egyptian Arabic that make it hard to transfer over NLP tools. We made an NLP library for Egyptian Arabic, including a tokenizer, MWT, POS tagger, NER, and constituency parser trained on short-form text. We then used these tools to train a question-answering model, creating a QA dataset by extracting clozes using constituency parser and answers using our POS tagger and NER, translating these into natural language questions via noisy cloze translation, and then fine-tuning a BERT MSA model on this Egyptian Arabic QA dataset to achieve an F1 score of 67.2 and EM score of 62.6 in named entity mention, which outperforms similar systems trained on English and supervised QA models trained on MSA.

1 Key Information to include

- Mentor: Irena Gao
- External Mentor: John Bauer
- Sharing Project: N/A

2 Introduction

Egyptian Arabic is the most commonly spoken dialect of Arabic, the first language of 110 million Egyptians, and the second most spoken dialect in many other countries in the Middle East, including Iraq, Jordan, and Libya. Despite the prevalence of Egyptian Arabic, there do not exist many NLP tools for Egyptian Arabic, where most Arabic NLP tools are trained using Modern Standard Arabic (MSA). Despite being the sixth most spoken language, there do not exist many datasets for downstream tasks in MSA, let alone Egyptian Arabic. Given the importance that Egyptian Arabic has in communication around the world, we created an NLP library for Egyptian Arabic, including a tokenizer, MWT, NER, and constituency parser.

To create the tools above, we used a constituency tree bank for Egyptian Arabic. However, for many other downstream tasks, such as extractive Question Answering, there are no datasets for Egyptian Arabic. This is due to the amount of time and effort it takes to manually generate questions and answers for text, meaning that this task is not easily automated. We used the tools from our library to create our own dataset for extractive question answering, extracting the proper nouns from sentences in a large corpus of Egyptian Arabic text and generating questions around those extracted answers. We did so by finding the sentences in our context paragraphs that contained these entities and translated them into natural language questions using a method called noisy cloze translation.

To demonstrate the performance that we could get on downstream tasks using our toolbox, we fine-tuned an Arabic (MSA) BERT model on the dataset compiled using these tools, with our model showing an F1 Score of 67.2 and EM score of 62.6, which outperforms a similar experiment conducted in English by Facebook AI mentioned in our related works. Because our dataset was compiled using a NER system, the majority of the questions tested were named entity mention questions.

There are 2 primary deliverables for this project: an NLP library for Egyptian Arabic and a QA dataset for Egyptian Arabic. We are looking to add our NLP library to Stanza (Qi et al., 2020) with the help of John Bauer. Looking at a supervised QA model from the American University of Beirut, when fine-tuning a BERT base model on a translated set of 48k question-answer pairs from SQuAD into MSA using machine translation and a crowd-sourced dataset of 1.3k question-answer pairs from Arabic Wikipedia, the highest F1 score reported was 48.6 (Mozannar et al., 2019). With there being more MSA data on Arabic Wikipedia and the performance of our model on just 94k question-answer pairs, we aim to expand this project in the future to create an extractive QA dataset for MSA with significantly more question-answer pairs and refine our methods of cloze translation to involve a UNMT, which we hope will produce more well-formed questions.

3 Related Work

To generate the QA dataset, we followed a similar architecture to a paper published by Facebook AI in 2019 that looked at how QA datasets could be created on a large corpus of English text (Lewis et al., 2019). In this work, an NER system and constituency parser were used to extract answers and the sentences or subclauses where they appear from paragraphs of texts. They then used two methods of cloze to natural language question translation, a rule-based approach named noisy cloze and a UNMT. The primary motivation behind this work was to explore methods of creating QA datasets that did not require manual extraction and question generation, similar to our motivations for creating a QA dataset for Egyptian Arabic, a largely underserved language in NLP tasks.

To produce results given the scope and timeframe for this project, we modified the rule-based noisy cloze approach that they used. Their method tokenized the given cloze, applied a noise function described in Lample et al. (2018) that consists of shuffle, dropout, and blank word layers, and prepends the wh* word corresponding to answer the using the labels from the NER system. This in turn generated natural language questions. However, when looking at the proportion of questions that were well-formed, between 0.8 and 2.7 of the questions were well-formed, dependent on the cloze extraction method used. Despite this limitation, a BERT-base model fine-tuned on the questions generated by noisy cloze achieved an F1 score of 46.1 and EM score of 36.5. These are both slightly lower than the scores from the questions generated by their UNMT, but due to the lack of complexity in the translation method, we opted to use this method when generating our dataset.

4 Approach

We have 2 main parts to our project, the NLP library for Egyptian Arabic and the unsupervised QA system trained on the dataset created using the NLP Library.

4.1 NLP Library

4.1.1 Tokenizer & MWT

For the tokenizer, we are using Stanford's Stanza NLP library (Qi et al., 2020) to train our model. We are using two constituency treebanks for Egyptian Arabic. Because we had constituency treebanks, everything in the datasets was already tokenized so we needed to extract the raw text and the tokens from this dataset. The datasets were composed of text messages and forum text. Each text conversation and article had its own xml file. In those xml files each sentence unit was between <su> tags but and each file had its own format of parent tags. There were 4 different formats that we manually discovered and then created 4 different ways to extract the text according to the format. Then to extract the tokens, we used a POS file that came with the dataset that had each token and its part of speech. We went through all the tokens and added them together in conNNL-U file format. We manually added periods to the sentences that didn't end with a punctuation mark so the tokenizer model can also learn to segment the sentences. Then, we discovered a discrepancy between the raw

file and the POS file that some of the letters in the raw text were not tokenized or written in the POS file at all. So, we went through the raw file and manually compiled a list of those letters that contained more than 130 letters all discovered manually. We first trained the tokenizer on only one dataset, but we figured out that the clitics were not correctly split sometimes, so we trained it on the two datasets combined, and from here on, we decided to train all the upcoming models on the two datasets combined.

4.1.2 Charlm and Word embeddings.

We used a dataset containing 2.5 million Egyptian sentences to train a character model. The character model we used was from Stanza and it is going to be used as part of the NER model, the POS tagger model, and the constituency parser model mentioned later. We then created our own word embeddings using the trained tokenizer to tokenize the 2.5 million sentences and then used them to create GloVE embeddings that had 50 dimensions. We used those embeddings and the embeddings from FastText to train the models discussed later, and our embeddings showed equal and sometimes better performance with only 50 dimensions compared to FastText which had 300 dimensions for each word.

4.1.3 NER

Then, to create the NER dataset, we needed to manually tag the sentences in our constituency treebank. To do so, we compiled all of the sentences from our treebanks into raw text and filtered the dataset we had previously to only include sentences that the treebank tagged as containing proper nouns from the POS file. We then gathered those different proper nouns and compiled a list of 1200 proper nouns used in the whole dataset. Then, we went through all of them classifying them either as person names, locations, time, organization, or misc. Then, for each type, we collected the sentences containing only that type of proper noun and tagged them with the S- along with their corresponding tag. Then, we collected the sentences that did have different types of nouns in them, nouns that needed more context to classify, or nouns that had nouns right next to each other to properly BIOES tag them. We tagged more than 2400 sentences manually using an interface that we built to expedite the process, shown in Figure 1. In the interface, the sentence was shown along with brackets [] around the proper nouns to easily identify them. This manual process gives us a gold standard NER dataset to train our model, again using Stanza. In total, we have over 8000 sentences containing proper nouns and other sentences from the dataset that didn't contain any proper nouns to create a dataset of more than 70000 data points for our NER model to train. From them, we created the JSON files required by the model.

Figure 1: NER BIOES Tagging Interface

4.1.4 Constituency Parser & POS-tagger

For our constituency parser, because the original dataset that we have is a constituency treebank, we can simply train our constituency parser on the original dataset using Stanza. Some of the constituency trees were poorly structured so we added a ROOT node in the beginning and recursively removed the NONE tags along with their parents if they have no siblings. Also, some of the trees had non-unary roots, so we had to add an S node as a parent to those nodes. Along with the constituency parser, we used the POS tags from the treebanks to train a POS tagger that we later utilized for our QA dataset. The tags in the POS file used were XPOS tags specifically created for the dataset. We manually went through the tags in 24 different groups of tags and created a dictionary that converts each group of XPOS tags into their universal UPOS tags. Again, we created the coNLL-U files and the mrg files for the POS tagger model and the constituency parser model respectively.

4.2 Unsupervised QA

4.2.1 Answer/Cloze Extraction & Translation

To generate the QA dataset, we used a list of 2.5 million sentences written in Egyptian Arabic. These included both short-form and long-form text, ranging from short text messages to entire articles written. Of these 2.5 million sentences, we generated a dataset of over 94k question-answer pairs. This is nearly double the largest QA dataset that we could find compiled on MSA, which used an English to Arabic NMT to translate 48k SQuAD datapoints (Mozannar et al., 2019).

For answer extraction, we were originally going to use the NER system developed from our toolkit. However, we did not have a high-performing NER when we needed to begin this part of the project, having an F1 score of 73. We later found what was causing the error in our NER system, which we discuss in section 5.3, but we pursued the following alternative for our answer and cloze extraction. Instead of directly extracting the entities from our dataset of 2.5 million sentences, we used our POS tagger to find all sentences that contained at least one proper noun and used only those data points. We then extracted just the proper nouns and used our NER to label them with BIOES tags to identify the type of answer and the type of the corresponding question, which it succeeded at, leaving us with 94k answers.

For cloze extraction, we used the entire sentence that the tagged answer had appeared in. This gave us significantly better clozes compared to our original plan of using the nearest subclause that the proper noun had appeared in, which we had used our constituency parser for. Unlike the experiment from Lewis et al. (2019) where using the closest subclause achieved the highest performance, we saw less information loss when we used the closest sentence as our cloze compared to the closest subclause. Since the constituency trees generated were already tokenized, we didn't need to retokenize them.

For cloze translation, we used the architecture described in section 3 of this paper and illustrated in Figure 2. This gave us a dataset with the following tuples of information: the context, the translated question, the extracted answer, and the indices of the start and end tokens.

4.2.2 Fine-tuning Arabic BERT

To train our QA system, we used the dataset produced from our cloze translations to fine-tune the Base BERT Arabic model Safaya et al. (2020). We were originally planning on using the XLNet pre-trained model, which has shown greater performance on QA tasks, but because there are not any XLNet embeddings for Arabic, we opted to use a pre-trained BERT model on primarily MSA and fine-tune it on our Egyptian Arabic QA dataset.

We explored the option of pre-training BERT on our corpus of 2.5 million sentences in Egyptian Arabic to produce better embeddings for this dialect of Arabic, but we could not do so due to time constraints, where the model was estimated to take around a week to finish training. For future work, we plan on fine-tuning our own Egyptian Arabic BERT on our QA dataset to see if there is a performance increase when trained on a single dialect of Arabic as opposed to the two different dialects separately.

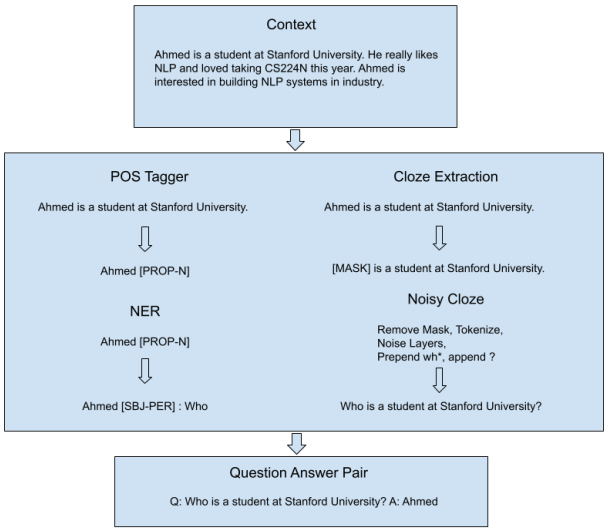

Figure 2: Cloze Translation Architecture (translated in English)

5 Experiments

5.1 Data

We have three datasets mentioned in our approach. Two are short-form sentences coming from text messages and forum posts, LDC2018T23: BOLT Egyptian Arabic Treebank - Discussion Forum Maamouri, Mohamed et al. (2018) and LDC2021T17: BOLT Egyptian Arabic Treebank - SMS/Chat Maamouri et al. (2021). They are constituency treebanks, meaning that we directly trained our constituency parser on them but we needed to manually adjust them for our other tasks. As described in our approach, we needed to manually tag each sentence for our NER system and format the treebanks for our tokenizers. The third is a compilation of 2.5 million sentences in Egyptian Arabic from Kaggle[1], that greatly differ in length. This is the dataset we used to generate our QA dataset

One exciting part of our project was the production of our own QA dataset for Egyptian Arabic. It is the largest open-source Arabic dataset, and the only dataset in the dialect. We are looking to improve the quality of the dataset in the future via other methods of cloze translation that may create more well-formed questions.

5.2 Evaluation method

For the evaluation metrics, we used Stanza's evaluation method which already gave us scores for each of the models we trained. For the word embeddings we created, they were evaluated based on the performance of the models compared to FastText embeddings.

For our QA dataset, we selected 100 question-answer pairs to gauge how well-formed the questions were. For the QA model, we used F1 and EM as our evaluation metrics.

5.3 Experimental details

We used the default settings from Stanza in most of the models, the tokenizer, along with the embedding creation took less than a day. The NER, POS, constituency parser, and character model, took around two days. We trained each model a couple of times to compare them and try to get different results. The tokenizer was trained on one dataset, then two datasets combined. The NER was trained one time with the CharLM and one without, then each one of those times we used FastText and another time we used our own embeddings. The POS was done the same way, except that we

[1]The dataset we used can be found here: `https://www.kaggle.com/datasets/mostafanofal/two-million-rows-egyptian-datasets`

didn't train it with FastText and without the CharLM due to time limitations. The constituency parser also was trained on the two datasets with the CharLM and without the CharLM.

One of the major changes in our project was the NER model. We initially has a systematic error in the dataset used that caused the NER model to perform poorly with an F1 score of around 73. After discovering the error, the dataset was fixed and the NER F1 score jumped to 88.5. The NER model trained on the new dataset was only trained using our own embeddings with the CharLM and without the CharLM. We didn't have time to train it with FastText.

5.4 Results

Tokenizer:

The tokenizer metrics are Tokens, Sentences, and Words scores. The tokens score is how accurately the tokens split by the tokenizer, including splitting the clitics.

Tokens	Sentences	Words
99.91	94.88	97.72

MWT:
Words Score (only metric from Stanza): 98.47

The POS:
The UPOS and the XPOS are the metrics for measuring the accuracy of predicting the universal parts of speeches in each word, and the specific parts of speech created for each word.

- With CharLM and FastText embeddings:

UPOS	XPOS
94.96	91.82

- With CharLM and Our own embeddings:

UPOS	XPOS
95.00	91.86

- Without CharLM and Our own embeddings:

UPOS	XPOS
94.67	91.42

NER with the old Dataset: We calculate the precision and recall then the F1 score for the model.

Custom Embeddings:

	Prec.	Rec.	F1
W/O CharLM	74.56	73.56	74.06
W CharLM	75.82	73.70	74.75

FastText Embeddings:

	Prec.	Rec.	F1
W/O CharLM	72.37	77.97	75.06
W CharLM	73.44	78.64	75.95

NER Using New Dataset & Custom Embeddings:

	Prec.	Rec.	F1
W/O CharLM	89.12	81.19	84.97
W CharLM	91.14	86.28	88.64

Constituency Parser:

	F1
Without CharLM	82.8
With CharLM	84.8

QA Model, all trained on BERT:

	EM	F1
Meta AI(English UQA)	30.3	39.5
Neural Arab (SQA)	34.1	48.6
Our Model	62.6	67.2

6 Analysis

Looking at our tokenizer, we saw that there was an imbalance between words that have clitics and words that do not, with the former being much more prevalent in the dataset. This causes words that should not be split into two different tokens to be split.

Looking at our Q&A model, we saw that it outperformed both the English unsupervised model developed at Facebook and the supervised model trained on MSA. We have two theories as to why our model outperformed them.

First, looking at the English unsupervised model, we noted earlier that a very small fraction of the questions formed by noisy cloze were well-formed. However, when looking over our generated dataset, we saw that significantly more of the questions generated in Arabic were well-formed, and approximately 33 percent of the questions we manually reviewed were well-formed. We believe that this is due to the noise function used, which caused the words to shuffle around in English, which changes the grammar of the question. However, in Arabic, you can change the order of some words in a question and still have it be grammatically correct, like having the wh* word in different areas of the sentence. This syntactic difference between the two languages made noisy cloze a better translation method in Arabic than it was in English. However, we noticed that it was shorter sentences that formed the best translations, so we are looking to improve our constituency parser to extract better subclauses, which in turn will generate better questions.

Looking at the supervised model train in MSA, we realized that there was significantly less data used when training. They crowd-sourced only 1.3k data points, which when compared to our 94k question-answer pairs explains the difference in performance that we saw. One interesting takeaway from their experiment is that when they trained on 48k question-answer pairs taken from SQuAD and translated into Arabic using an NMT, it performed worse than the model trained on just the 1.3k manually annotated data points. This indicates the quality of our dataset, given that using this cloze translation method on raw Arabic text performed better than a translated professional Q&A dataset.

7 Conclusion

Egyptian Arabic is the most spoken Arabic dialect, but an underserved language in NLP. Because there were none, we developed a whole NLP toolkit and used it to create the first open-source Egyptian Q&A dataset. The model performed really well, as well as the rest of the toolkit and we're preparing to put them officially into Stanza. There is still a lot left to do though. The tokenizer dataset should be improved to include more words that have clitics in them to be a balanced dataset. The POS would need to be trained with the new tokenizer to get more accurate word tagging. Then the NER dataset should also be larger and more balanced. We're also planning on manually adding different categories other than locations, organizations, person names, and times.

We also want to collect a larger dataset than the 2.5 million sentences dataset to train a BERT for Egyptian Arabic. We are planning on scraping more recent Facebook, Twitter, and Reddit data. Looking at how we can improve the Q&A dataset, it needs to be larger and manually revised. Finally, the most important part of the project is to create a universal dependency dataset for Egyptian Arabic. For each one of these tasks, we're going to crowd-source people and try to get funding so we can outsource this manual work to native Egyptians.

References

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based neural unsupervised machine translation.

Patrick S. H. Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. *CoRR*, abs/1906.04980.

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Dalila Tabassi, and Michael Ciul. 2021. Bolt egyptian arabic treebank - sms/chat.

Maamouri, Mohamed, Bies, Ann, Kulick, Seth, Krouna, Sondos, Tabassi, Dalila, and Ciul, Michael. 2018. Bolt egyptian arabic treebank - discussion forum.

Hussein Mozannar, Karl El Hajal, Elie Maamary, and Hazem Hajj. 2019. Neural arabic question answering.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059, Barcelona (online). International Committee for Computational Linguistics.