Improving English to Arabic Machine Translation

Wael Abid
Department of Computer Science
Stanford University
waelabid@stanford.edu

Younes Bensouda Mourri
Department of Statistics
Stanford University
younes@stanford.edu

Abstract

This paper implements a new architecture of the Transformers to translate English into Arabic. The paper also explores other modeling problems to further improve the results obtained on the English-Arabic Neural Machine Translation task. In order to correctly evaluate our models, we first run a few baselines, notably a word-based model, a character-based model and a vanilla Transformer. We then build on top of it by changing the architecture, using pretrained-embeddings and modifying the morphology of the Arabic language tokens. We note that the best model we got, weighing both training time and metric evaluation, used a variation of the transformers with morpholgy modification and pretrained embeddings. We then do ablative analysis and error analysis to see how much improvement was made by each addition to the model.

1 Introduction

Arabic to English and English to Arabic translation is not very well explored in the literature due to the lack of a very big and varied corpus of data. Arabic is a difficult language to master and there aren't enough NLP researchers working on the matter to make it as developed as English NLP research.
The potential bottleneck behind such research is the understanding of the linguistic structure of the Arabic language. Arabic is a morphologically rich language and usually combines pronouns, conjugation, and gender in one word. For example, the word ولمدرستها (walimadrasatiha) is one word. However in some cases each letter represents a word. The prefix و (wa) corresponds to and, the letter ل (li) corresponds to the word for, مدرست (madrasa) means school, and the suffix ها (ha) corresponds to the gender pronoun 'her'. Hence, even when computing the BLEU score, one very small suffix could easily lower your overall results although you got the other three words right.
These complexities have made Arabic machine translation difficult to improve on. To add to these complexities, the same word could mean very different things depending on how it is diacritized. Diacritization is the addition of short vowels to each word which changes both the pronounciation. This means that in some cases, even though two words can be written with the same letter but could mean two completely different things.
Furthermore, Arabic is a low resource language and there isn't a lot of data out there to train big and models that can represent the complexity of the language. To solve this problem, we claim that using pre-trained embeddings and modifying the morphology of the words by expressing each word in its sub-words will help. Since Arabic requires many layers of abstractions that are similar due to its morphological structure, we believe that the concatenation of the hidden layers prior to the projection layer in the multi-headed self attention part is not necessary, and we believe that shared weights in that layer will be enough because. This will be further analyzed in the Analysis part of this report.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2 Related work

One rather interesting paper on Arabic machine translation is the "Triangular Architecture for Rare Language Translation"[1] (Ren. S 2018). This paper trains English to Arabic by using a triangular method. It first trains English to French and then uses the well translated corpus as the new target. It then translates English to Arabic and Arabic to French. In doing so, you can use the rich language resources/labelled data to solve the problem of a low resource language. As a result, they got better results for both the English to Arabic and the Arabic to French translations.

Another paper "Transfer Learning for Low-Resource Neural Machine Translation"[2] (Zoph. B 2016), that didn't necessarily target the English-Arabic task, used transfer learning and got an improvement in their BLEU in low-resource machine translation task.

A paper "When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation?"[3] (Qi. Y 2018) that used pretrained embeddings had an improvement of their BLEU score as well.

Other people are working on the morphological structure of the Arabic language. A paper called "Orthographic and morphological processing for English–Arabic statistical machine translation"[4] (El Kholy 2012) explores morphological tokenization and orthographic normalization techniques to improve machine translation tasks of morphologically rich languages, notably Arabic. The two main implementation of Arabic segementation are "Farasa: A Fast and Furious Segmenter for Arabic"[5] (Abdelali et al. 2016) and "MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic"[6] (Pasha et al. 2014). These two systems for morphological analysis and disambiguation of Arabic and segementation.

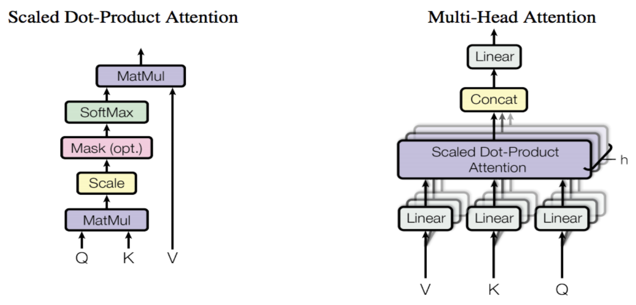
Another paper is "Arabic-English Parallel Corpus: A New Resource for Translation Training and Language Teaching"[7] (Alotaibi 2017). It explores the different data-sets that can be used for the problem as well as their types.

"The AMARA Corpus: Building Resources for Translating the Web's Educational Content"[8] (Guzman 2013) presents TED talks parallel data, and "A Parallel Corpus for Evaluating Machine Translation between Arabic and European Languages" presents the Arab-Acquis data of the European Parliment proceedings. In addition to this "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles"[9] (Lison 2016) presents parallel data of movie subtitles.

3 Approach

The baseline model is the character-based model level encoding for the Arabic corpus. For each character, we would look up an index and get the embedding. We would then convolve a filter around the character embeddings and pass it through a max-pool layer and then use a highway network with a skip connection to combine these into an embedding that represents the word and after that we apply our dropout layer. Description of the original contributions to these baseline models are described in the experiments section.

We then used a transformer model as described in "Attention is All[10] (Vaswani 2017) from OpenNMT as our vanilla Transformer model that we will improve later, as we approached both architectural and modeling problems of the model applied to our task. After a challenging amount of pre-processing and preparation of the data pipeline, we ran the model to get a baseline score. As a refresher, the transformer network is much faster the the normal seq2seq RNN because it allows for parrallel compuatation. Its structure was designed so that at each time when predicting, the model has access to all the positional encodings of each word. The best way to understand the transformers is to think of them as two stacks. The first stack is the encoder which consists of several units. Each unit has a multi-headed attention followed by some normalization layer and a skip connection. The output is then followed by a Feed-forward and another normalization layer. This is considered one unit and there are N of these in the encoder. We will first explain the multi-headed self attention as described by the transformers paper and then we will explain our new architecture. The image below describes the mult-headed self attention in comparison to the normal scaledwhich is the meet of this new paper:



The first image to the left could be described with equations as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

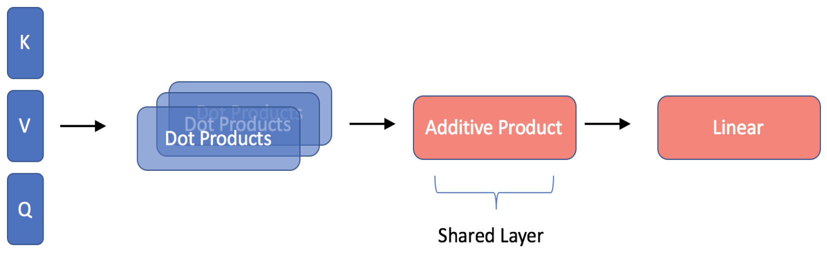
Q stands for the query, K stands for the keys, and V stands for the values. The larger the query times the key is the more attention will be places on that key. The intuition stems from similar encodings tend to have higher dot products. The multi-headed self attention is slightly different.

$$MultiHead(Q, K, V) = Concat\left(head_1, \ldots, head_h\right)W^O$$

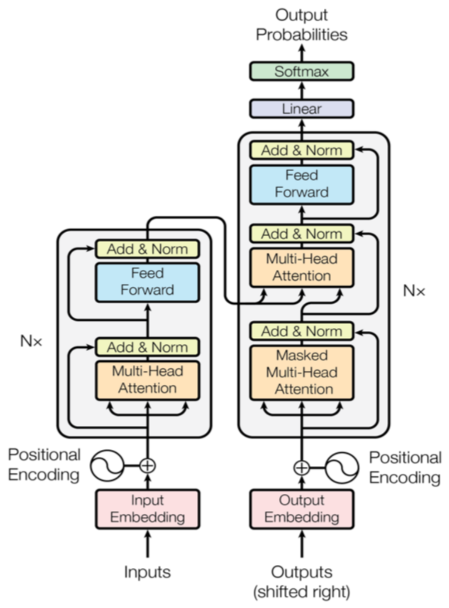
where $head_i$ is

$$Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

In our new architecture we use the following structure for the multi-head self attention. We create a shared embedding after all the heads and use that as a projection to W^O. This gives us way fewer parameters. Our new multi-self headed attention is as follows:



The goal of adding the shared layer between all the previous dot products allows us to have a shared layer which speeds up the computation of the W^O matrix described in the paper. After modifying the multi-headed self we proceed with the following model.



The model above has two main stacks: the encoder and the decoder. When decoding, we use a mechanism so that we not only look at all the previous positions of the output, but we also look at all the input. This gives way better results. The feed forward layer is defined as:

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

Where the Ws are weight matrices used for the projections. We also used pre-trained embeddings and transfer learning which greatly improved our model as discussed in the experiments section.

4 Experiments

4.1 Data

We compiled our data from different sources and domains so that the model doesn't learn a specific language or writing style, and that it learns both formal and less formal Arabic (Modern Standard Arabic, not colloquial).
We used the Arab-Acquis data as described in "A Parallel Corpus for Evaluating Machine Translation between Arabic and European Languages" which presents the Arab-Acquis data of the European Parliment proceedings totalling over 600,000 words. We combine that with the dataset in the paper "The AMARA Corpus: Building Resources for Translating the Web's Educational Content"[8] (Guzman 2013) which consists of TED talks parallel data totalling nearly 2.6M words. In addition to this we add 1M words of "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles"[9] (Lison 2016) movie subtitles movie data.

We found that data was not perfectly parallel in terms of number of lines and we had to know where the shift between the number of lines is. Therefore, we wrote a program that detects those descrepancies, and we went on to fix the error directly in the files.
Since our dataset is composed of 4 different sources, there was some formatting differences. For example, some files had to have a full stop at the end of the file in a single line to signify the end of the file while some didn't. Some data-sets came in one file, and some came in hundreds of files, so we had to concatenate while accounting for the formatting differences and making sure everything is uniform both in the Arabic files and the English files.
For the low resource experiment, we used the Arab-acquis dataset. The data-set referred to above has a parallel corpus of over 12,000 sentences from the JRCAcquis (Acquis Communautaire) corpus translated twice by professional translators, once from English and once from French, and totaling over 600,000 words. We used it because it's small and very high quality whereas the other datasets

didn't have these qualities. The split is 80 train, 10 dev, 10 test.
In the character-based model, we had to edit some of the code we already wrote in the assignments to adapt it to Arabic characters, specifically make sure that char2id and id2char reflect the totality of our characters. One of the major things is that we kept into running into key errors a lot and had to debug that and make sure that every character of both languages is represented in the dictionary. One other thing is that it seemed that the data needed some other pre-processing because it had some special characters like the "right to left" character and the "left to right" character. Another concern in the beginning was that the Arabic characters were not showing in their "natural/normal" states but they were showing in their respective unicodes (e.g. u0630). That happened in the vocab.json file but it didn't seem to matter that much.
However, in the Transformer model, we didn't have to do this because the data pipeline was language agnostic

4.2 Evaluation Method

BLEU score as a metric uses strict word and phrase matching between the MT output and reference translations. For morphologically rich target languages such as Arabic, such criteria are too simplistic and inadequate.

"A Human Judgment Corpus and a Metric for Arabic MT Evaluation"[11] (Bouamor 2014) proposes the Arabic Language BLEU (AL-BLEU) metric which extends BLEU to deal with Arabic rich morphology. They extend the matching to morphological, syntactic and lexical levels with an optimized partial credit. AL-BLEU starts with the exact matching of hypothesis tokens against the reference tokens. Furthermore, it considers the following: (a) morphological and syntactic feature matching, (b) stem matching. Based on Arabic linguistic intuition, we check the matching of a subset of 5 morphological features: (i) POS tag, (ii) gender (iii) number (iv) person (v) definiteness.
Although we weren't able to obtain the implementation of AL-BLEU, we replicated some of the logic they discuss. What we did is instead of applying BLEU score on the original Arabic text itself, we apply BLEU to the tokenized form of the same text to get a score that is closer to the human judgement.

4.3 Experimental Details and results

The first step of the project was adapting our implementation of the character-based model (A) from class to our English-Arabic task. After running into CUDA error memory space, we had to lower the batch size from 32 to 16, which had benefited the performance. Bringing the batch size to 24 made it worse than 16. What helped a lot and boosted our performance is making the max epoch equal to 75 instead of 30 making it train for longer. We also attempted to do stacked LSTM decoder but that drastically lowered our performance so we refrained. What also helped is making the kernel size smaller, from 5 to 4. We believe this helped because Arabic encapsulates more meaning in less words than English. The end model took 13.5 hours to train.

Later, we moved to the transformer model. We used the implementation of OpenNMT[12] (Klein 2017) as a second baseline (B). This model's data pipeline was more approachable than the first baseline model, and we just needed to work on our data cleaning, parallelism and uniformity to get it to train.
This model improved on the other baseline model by 2.27 BLEU points and it took 8.5 hours to train. We therefore picked the transformer model to work with and improve our results with modeling problems and architecture problems.

The transformer baseline was trained on the raw text (i.e. words are in their original morphological state). However, for motives that will be explained later in the Analysis part, we chose to process our raw text data and tokenize it to train our model and obtain better results. We used MADAMIRA[6] (Pasha 2014) to tokenize the text and separate subwords that pertain to (i) gender (ii) number (iii) person (iv) definiteness.

In a first experiment (I), all of the training, validation and training test were tokenized, but in a later experiment (II) we had raw training and validation data, but tokenized the test data. The goal of these two experiments in to see if the transformer model can recognize the patterns of (i) gender (ii) number (iii) person (iv) definiteness better if these patterns were separate or compound in one word. The model that performed better is the one (C) from experiment (I) where all the data was tokenized.

It performed 2.52 BLEU point better than the model (D) from experiment (II). However, the model (D) from experiment (II) outperformed the vanilla transformer model (B), which is trained and tested on raw data, by 0.54 BLEU points.

The last iteration of modeling experiments was for a low resource translation task, which consisted of training a model (E) with pre-trained embeddings. We obtained an improvement of 2.74 BLEU points compared to the same model (C) trained minus the pre-trained embeddings.

From an architectural point of view, we implemented the transformer model modification discussed in the Approach section. This model (F) took 10 percent less time to train but yielded a 1.35-BLEU point lower score. Yet, we believe that this loss in BLEU score is not a big compromise to obtain 10 percent faster training.

The scores of each model are presented in the following table:

Model	BLEU
(A) Baseline character-based	28.46
(B) Baseline Transformer	30.73
(C) All Tokenized Model	33.79
(D) Test Only Tokenized Model	31.27
(E) All Tokenized + Pre-Trained Embeddings Model	36.53
(F) Muti-Head Shared Attention	35.18

5 Analysis

The Transformer outperforms the character-based model because it only performs a small, constant number of steps (chosen empirically)and in each step, it applies a self-attention mechanism which directly models relationships between all words in a sentence, regardless of their respective position and this makes it possible to preserve long-range dependencies in a sentence and make good predictions of the next word.

The reason why we think morpholgy-aware tokenization works is the following: as previously stated, Arabic is a morphologically rich language, and this makes Arabic Neural Machine Translation difficult to approach as it increases the number of out-of-vocabulary tokens which means that it consequently worsens the issue of data sparsity. One other problem is that it makes it difficult to have parallel words in the two different languages (in our case it's Arabic and English) which makes it more difficult to get good translations. Tokenization solves this problem and makes for a more parallel data between English and Arabic.

We add to this pre-trained embeddings which improve our results very well, and we believe this is due to the fact that these pre-trained embeddings have been part of a training process where they were trained on very large datasets which makes them encapsulate a lot of the meaning and context of those words that are otherwise not frequent in our low-resource pair.

As for weight sharing, the logic behind it is that repeated patterns are everywhere in languages and Arabic specifically. For example, in a sentence where the gender of the subject is feminine, the suffix ها (ha) will be repeater again and again. This means that if you see just part of the word it would be enough to make a conclusion. Another idea is the number of parameters will be less and it will speed up the learning. We believe that we do not need an entire matrix of parameters to identify a task. You could try to make use of repeated patterns (or general ones in this case).
We modified the architecture of the transformer model because we believe that the repeated patterns in language would translate into repeated patterns in the multi-headed self attention that could be learned simultaneously. We can see the benefits of such distributed attention, but we believe that it is overkill to implement it all the way, especially that it only slightly reduces the BLEU score. This means that we can effectively reduce the number of parameters by [heads*(model dimension - 1] parameters and still keep the same performance. As a result we do speed up the training and reduce model complexity. Hence, we consider that our hypothesis of a shared layer after the key/value/query product has proven to be effective.

6 Conclusion and Future Work

We present English to Arabic Neural Machine Translation on a rich and diverse dataset including a first extended result on English to Arabic NMT using Transformers. We show the importance of morphology-aware preprocessing and pretrained embeddings and how they contribute to a better translation.

The next steps of this project and for Arabic NLP in general are working on a new metric. BLEU score isn't well-suited for morphologically rich languages, notably Arabic. BLEU score is too harsh a metric even for other languages and doesn't capture the true meaning of a sentence. We think that a good metric would be one that focuses on the morphological components of the words rather than the exact form of it. One other important step towards improving Arabic machine translation and other NLP tasks is better morphologically-aware tokenization as FARASA[5] (Abdelali 2016) and MADAMIRA[6] (Pasha 2014) only provide an accuracy of 84 percent.

References

[1] Shuo Ren, Wenhu Chen, Shujie Liu, Mu Li, Ming Zhou, and Shuai Ma. Triangular architecture for rare language translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 56–65. Association for Computational Linguistics, 2018.

[2] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575. Association for Computational Linguistics, 2016.

[3] Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[4] Ahmed El Kholy and Nizar Habash. Orthographic and morphological processing for english–arabic statistical machine translation. *Machine Translation*, 26(1):25–45, Mar 2012.

[5] Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. Farasa: A fast and furious segmenter for arabic. In *HLT-NAACL Demos*, 2016.

[6] Arfath Pasha, Mohamed Al-Badrashiny, Mona T. Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC*, 2014.

[7] Hind Alotaibi. Arabic-english parallel corpus: A new resource for translation training and language teaching. *Arab World English Journal*, 8:319–337, 09 2017.

[8] Francisco Guzman, Hassan Sajjad, Stephan Vogel, and Ahmed Abdelali. The amara corpus: Building resources for translating the web's educational content. 12 2013.

[9] Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC*, 2016.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[11] Houda Bouamor, Hanan Alshikhabobakr, Behrang Mohit, and Kemal Oflazer. A human judgement corpus and a metric for arabic mt evaluation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 207–213. Association for Computational Linguistics, 2014.

[12] Guillaume Klein, Yoon Kim, Yuntian Deng, Josep Maria Crego, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *ACL*, 2017.