ARBML: Democratizing Arabic Natural Language Processing Tools

Zaid Alyafeai
Dhahran, Saudi Arabia
g201080740@kfupm.edu.sa

Maged S. Al-Shaibani
Dhahran, Saudi Arabia
g201381710@kfupm.edu.sa

Abstract

Automating natural language understanding is a lifelong quest addressed for decades. With the help of advances in machine learning and particularly, deep learning, we are able to produce state of the art models that can imitate human interactions with languages. Unfortunately, these advances are controlled by the availability of language resources. Arabic advances in this field , although it has a great potential, are still limited. This is apparent in both research and development. In this paper, we showcase some NLP models we trained for Arabic. We also present our methodology and pipeline to build such models from data collection, data preprocessing, tokenization and model deployment. These tools help in the advancement of the field and provide a systematic approach for extending NLP tools to many languages.

1 Introduction

Arabic language is a widely used language. It is the sixth most spoken language in the world (Farghaly and Shaalan, 2009). It also has a noticeable influence on many other languages around the globe. Compared to English, Arabic is morphologically a very rich language (Habash, 2010) with relatively complex grammar and cursive script including the use of diacritics. Diacritics are special characters added to Arabic writing to replace the absence of short vowels. Moreover, Arabic has a variety of dialects that may greatly differ in style and grammar.

Arabic content on the web is vastly emerging with great diversity in style and subjects, written in many dialects. This opportunity opens doors for research to hone machine capabilities to automate language understanding and comprehension. However, Arabic inherent characteristics makes it difficult to resolve and require linguistic expertise.

Natural Language Processing is gaining a lot of attractions within the research community. The aim is to create machines that can replicate or exceed human language understanding. On another perspective, a lot of effort is invested to develop software applications to port research advances to industry. Another effort is also directed to facilitate researchers job by automating routine workflows, cleaning and preprocessing, for example. Some examples of this are huggingface, allennlp and flare. Most of these tools are designed to work for English or generalize the pipeline to work for other languages. Arabic, although it is not as popular as other languages tools, also has some contributions, but in the linguistics part only. Some promising examples are MADAMIRA (Pasha et al., 2014), FARASA (Abdelali et al., 2016), Adawat (Zerrouki, 2020) and CAMeL NLP (Obeid et al., 2020). These tools address a large spectrum of NLP tasks for Arabic like segmentation, part of speech tagging, named entity recognition, diacritizatoin and grammatical analysis. However, most of these tools are not using the recent advances in NLP. Unfortunately, in the Arabic community, open source contribution is not widely accepted. This is due to the copyrights restrictions made by authors as some of these tools are not licensed for commercial use. Although, the source code can be delivered on demand, this mechanism is still not development friendly with unclear methodology and processes to version control and collaboration.

In this paper, we introduce our contribution to the Arabic language open source community. We present a collection of models that can be utilized and improved to solve a wide variety of many Natural Language Processing tasks. Moreover, we introduce three libraries for scrapping, cleaning and tokenization. We also provide notebooks that can be easily used to replicate our experiments. We provide a flexible code design that can be implemented and extended to other languages.

Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS), pages 8–13

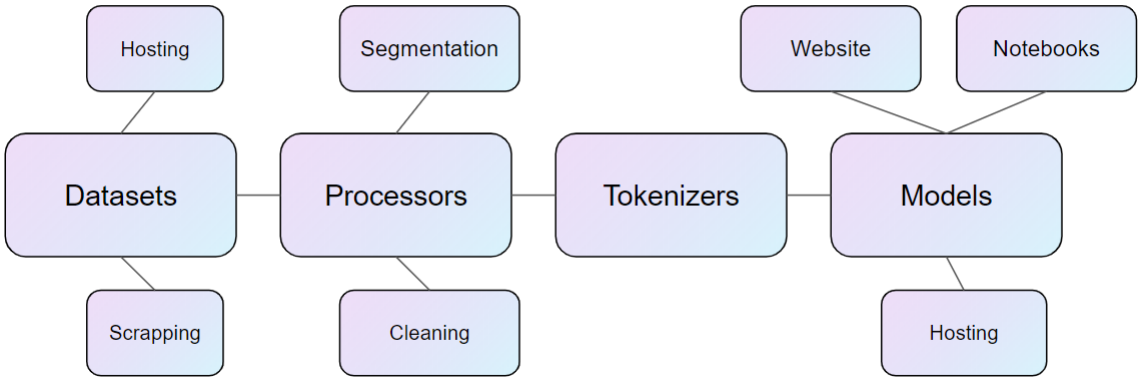

Figure 1: ARBML pipeline.

2 Design

We created ARBML in the hope of democratizing Arabic natural language processing by creating a set of demos as well as tools for making it easy to use for novice users, developers and researchers. We revise the NLP pipeline and make it suitable for Arabic as indicated in Figure 1. We provide datasets, preprocessors, tokenizers and models. Furthermore, we host notebooks that can replicate our experiments and help users to understand how to do each task. In the next few sections we explain our main tools.

2.1 Tnqeeb

This is a repository that hosts a collections of data gathered from multiple websites. This data is collected using scrapy, a well-known python scraping library. This implementation comes as a mature result after a sequence of scraping efforts using low level python libraries like *requests* and *beautifulsoup*. The current available data is a collected from three giant Arabic poetry websites: aldiwan, poetry encyclopedia, and poets gate. We plan to scrape as many sources as possible on a given topic, poetry, news, or blogs for instance. We then group, do initial processing, and de-duplicate these data into an individual repositories to be easy to work on.

2.2 Tnkeeh

Tnkeeh is a library that is responsible for preprocessing datasets. It has four main procedures

- Cleaning: this module is used for cleaning datasets by removing diacritics, extra spaces, remove English characters and remove *Tatweel* - a character used for increasing the length of characters.
- Segmentation: we use FARASA (Abdelali et al., 2016) for segmenting texts into morphemes.
- Normalization: Arabic letters can appear in different forms due to different Unicode's representing the same characters. We created a dictionary to map the same representations to their fixed characters.
- Data Splitting: we use a set of of procedures to split different types of datasets depending on the tasks to train on. For instance, we can split datasets if they are for unsupervised, classifications or parallel tasks.
- Reading: this module reads the different modes of datasets into python variables.

2.3 Tkseem

Tkseem is a tokenization library that implements multiple tokenization algorithms optimized for Arabic. We provide six categories of tokenizers with a simple interface.

- Word Tokenizer: splits words based on white spaces.
- Character Tokenizer: splits characters depending on their position on text.
- Sentencepiece Tokenizer: A wrapper for the sentencepiece library (Kudo and Richardson, 2018).

Dataset	Description
Arabic Digits	70,000 images (28 x 28) (El-Sawy et al., 2016)
Arabic Letters	16,759 images (32 x 32) (El-Sawy et al., 2016)
Arabic Poems	146,604 poems scrapped from (Aldiwan, 2013)
Arabic Translation	100,000 parallel Arabic to English translation ported from OpenSubtitles
Product Reviews	1,648 reviews on products ported from (ElSahar and El-Beltagy, 2015)
Image Captions	30,000 Image paths with captions extracted and translated from (Lin et al., 2014)
Arabic Poem Meters	55,440 verses with their associated meters collected from (Aldiwan, 2013)
Arabic Fonts	516 images (100 x100) for two fonts

Table 1: Collected and preprocessed Datasets.

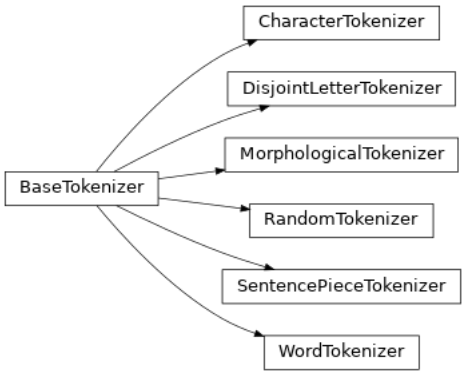

Figure 2: Base Tokenizer.

Function	Description
`tokenize`	Converts text to tokens
`encode`	Converts tokens to integers
`decode`	Converts integers to tokens
`detokenize`	Converts tokens back to text
`encode_sentences`	Encodes a set of sentences
`load_model`	Loads a saved model
`save_model`	Saves a given model
`encode_and_save`	Encodes and saves the model as numpy array

Table 2: Tokenizer functions.

- Morphological Tokenizer: splits words based on morphology. This was trained using Madamira (Pasha et al., 2014) on a large Arabic news corpus.
- Random Tokenizer: tokenizes text based on random splitting of words.
- Disjoint Letter Tokenizer: splits based on letters that are not connected in Arabic script writing.

All these tokenizers extend a common Base class Tokenizer (Figure 2) that implements the main functionalities like *encode*, *decode*, *tokenize*, and *detokenize* (Table 2). One useful function of these tokenizers is the ability to serialize them and load them on demand. This approach relaxes the time for training specially on large corpus. We also provide different methods for optimization like caching and memory-mapped files to speed up the tokenization process.

These tokenizers are evaluated on three NLP tasks: sentiment analysis, Arabic meter poetry classification and neural machine translation.

2.4 Models and Datasets

This main module is responsible for storing and serving different datasets and models. The main purpose is to give a real time experience for different models that are related to Arabic and NLP. The main strategy is highlighted in Figure 3. This procedure shows off our main approach for making the models easily accessible via different interfaces. We follow three main stages

- Preprocess Dataset: we collect and preprocess different datasets that are related to different tasks. Table 1 shows the main datasets that we collected. The datasets cover different areas like translation, sentiment analysis, poem classification, etc.
- Training: We train the model on the datasets

Model	Description
Arabic Diacritization	Simple RNN model ported from (Barqawi, 2017)
Arabic2English Translation	seq2seq with Attention
Arabic Poem Generation	CharRNN model with multinomial distribution
Arabic Words Embedding	N-Grams model ported from Aravec (Soliman et al., 2017)
Arabic Sentiment Classification	RNN with Bidirectional layer
Arabic Image Captioning	Encoder-Decoder architecture with attention
Arabic Word Similarity	Embedding layers using cosine similarity
Arabic Digits Classification	Basic RNN model with classification head
Arabic Speech Recognition	Basic signal processing and classification
Arabic Object Detection	SSD Object detection model
Arabic Poems Meter Classification	Bidirectional GRU from (Al-shaibani et al., 2020)
Arabic Font Classification	CNN

Table 3: Trained and deployed models.

using Keras with TensorFlow backend (Abadi et al., 2016). We used Keras because it is straight forward to convert the models using TensorFlow.js. We use Google Colab for training our models with proper documentations in a tutorial-like procedure. We make all the model available in this repository.

- Deployment: We make the models available in the browser using TensorFlow.js (Smilkov et al., 2019). TensorFlow.js is part of the TensorFlow ecosystem that supports training and inference of machine learning models in the browser. The main advantage is a device-agnostic approach that makes all the models available on any device that has a browser. Moreover, the models are light and can run offline. The main motive is to make the models easily accessible via a simple interface like the browser. This makes it easier for users to test different models in a few clicks.

We make all the datasets and models available on our GitHub : https://github.com/ARBML/ARBML. The procedure we follow makes it easier for developers to contribute to our project. Moreover, our strategy is language-agnostic and encourages extending it to other languages. In Table 4 we compare ARBML against other tools in the literature.

3 End-user Experience

ARBML provides a solid contribution from two main perspectives:

- Educational Perspective: People who wish to learn NLP will greatly benefit from ARBML.

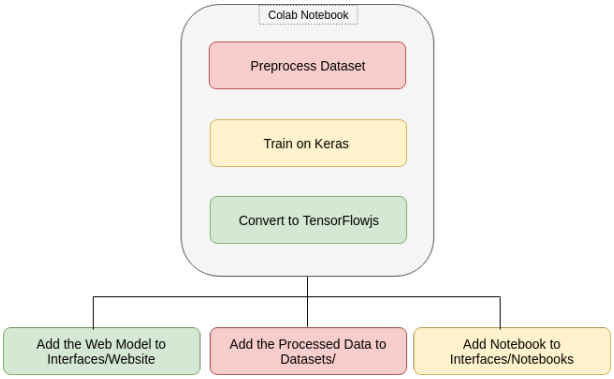

Figure 3: Models procedure.

ARBML provides various training models with different techniques for both, command line and web interfaces using Keras and TensorFlow.js. The pipeline from cleaning the dataset to model training and deployment is documented in details as Colab Notebooks. Additionally, users can test different models directly in the browser. For instance, we have a translation model where the users can enter a statement in Arabic and it will be translated in English.

- Development and Research: From development prospective a lot of tools like tnqeeb, tkseem and tnkheeh can be used in different projects related to NLP. Furthermore, developers can use our deployed models as prototype to test the possibility of implementing them in industry. Moreover researchers can use our pipeline to create new state-of-the-art models by using our models as a starting point.

Framework	Methodology	Models Availability	Datasets	Accessibility	Generalization	Programming Language	Web
Stanford CoreNLP (Manning et al., 2014)	Probabilistic models	N/A	N/A	The source code can be downloaded from the main website	Multilingual	Java	No
MADAMIRA (Pasha et al., 2014)	Morphological analysis developed with machine learning tools	Models are available via the API and the web interface	N/A	Source code is accessible by email for education uses only	Arabic	Java	Yes
FARASA (Abdelali et al., 2016)	Different Models built with machine learning	Binaries are available but the code used for training the models is not	N/A	Source code for the API is accessible by email with license for educational uses only	Arabic	Java	Yes
CAMeL NLP (Obeid et al., 2020)	Multitasks learning models	Models are available for download	Yes	Source code is open on GitHub	Arabic	Python	No
Adawat (Zerrouki, 2020)	General tools not necessarily built with machine learning	N/A	Yes	Source code is open on GitHub	Arabic	Python	Yes
ARBML	NLP tasks built with the recent advances in NLP field	Models are available on GitHub	Yes	Source code is available on GitHub along with the training notebooks	Generalizable	Python	Yes

Table 4: Comparing ARBML against other Arabic NLP tools.

4 Conclusion and future plans

Recently, many NLP tools have been developed but they only focus on English. In this work we showcased ARBML which is a set of tools that make Arabic NLP easily accessible through different interfaces. We target the NLP pipeline starting from scrapping datasets, preprocessing, tokenization to training and deployment. We focused on making the design of our tools language-agnostic and hence can be extended to many other languages, given we change the morphological aspects. We collected many datasets that can be easily used by researchers to develop new models. We also designed three libraries tnqeeb, tnkeeh and tkseem which can be easily utilized by developers to develop tools to support Arabic NLP. The tools utilize the morphological nature of Arabic to provide different functionalities that are unique for Arabic.

We plan to add many other models and make them easily accessible through the browser. Mainly, our next step is to tackle more advanced models like transformers (Vaswani et al., 2017). Furthermore, we want to apply different techniques like quantization and distillation to make the models available in the browser. Moreover, we would like to focus on light models like MobileBERT (Sun et al., 2020), retrain it for Arabic and make it readily usable in the browser.

5 Acknowledgements

ARBML is and open source project that will keep growing in the future. We would like to thank all developers who shared ideas, models and helped us address different issues.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 11–16.

Maged S Al-shaibani, Zaid Alyafeai, and Irfan Ahmad. 2020. Meter classification of arabic poems using deep bidirectional recurrent neural networks. *Pattern Recognition Letters*.

Aldiwan. 2013. Aldiwan.

Zerrouki Barqawi. 2017. Shakkala, arabic text vocalization.

Ahmed El-Sawy, EL-Bakry Hazem, and Mohamed Loey. 2016. Cnn for handwritten arabic digits recognition based on lenet-5. In *International conference on advanced intelligent systems and informatics*, pages 566–575. Springer.

Hady ElSahar and Samhaa R El-Beltagy. 2015. Building large arabic multi-domain resources for sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 23–34. Springer.

Ali Farghaly and Khaled Shaalan. 2009. Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):1–22.

Nizar Y Habash. 2010. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. Camel tools: An open source python toolkit for arabic natural language processing. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 7022–7032.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Lrec*, volume 14, pages 1094–1101.

Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. 2019. Tensorflow. js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350*.

Abu Bakr Soliman, Kareem Eissa, and Samhaa R El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Taha Zerrouki. 2020. Towards an open platform for arabic language processing.