Arabic Part of Speech Tagger

Aisha M.Badrawy, Dr.Mohamed E.Mahdy
German Univeristy in Cairo, Cairo, Egypt
aisha.badrawy@gmail.com
melmahdy@gmail.com

IndabaX Egypt 2019
https://sites.google.com/view/indabaxegypt

EGYPT
IndabaX

Introduction and Motivation

Arabic is a language spoken by around 400 million people, making it one of the most spoken languages in the world. It is the official language in 22 countries, dominantly lying in the middle east and north Africa.

Many of the applications that we use on a daily basis incorporate NLP, from simple tasks such as automatic text correction to speech recognition.

Still research interest in processing Arabic language compared to English language is very limited. Thus, the purpose of this paper is to implement a supervised Arabic Part of Speech (POS) Tagger, as POS tagging is an important task in NLP, that serves as a building blockformore advancedtasks.

Background

• POS Tagging is the problem of assigning tags, using a defined Tag set to un-tagged new words given a previously tagged Corpus.
• A tagset can be as general as (Noun, Verb, Adverb) or as detailed as (male or female, singular or plural, past or present).

Major Contributions

• Implemented a Neural Supervised Arabic Part of speech tagger that achieved state of the art results.
• Experimented with various machine learning algorithms namely Naïve Bayes, Support Vector Machines, Random Forest and Neural Networks.
•Trained a word2vec model for Arabic and Obtained Arabic word embedding that were used to train the models.
• Implemented a Neural Network with sequence to sequence architecture for Arabic POS.

Methodology

Dataset

• The data set used is the Prague Arabic Dependency Treebank (PADT).
• It is comprised of news texts and articles from 6 sources.

Data processing:

•The PADT tags were reduced to to the most common tags according universal tagset.
•PADT tags were reduced from 21 tags to 11 tags.
•Tokens were filtered, by removing unknown words, punctuations and numerals.
•The final shape of the data is 8 tags and 42,632 tokens.

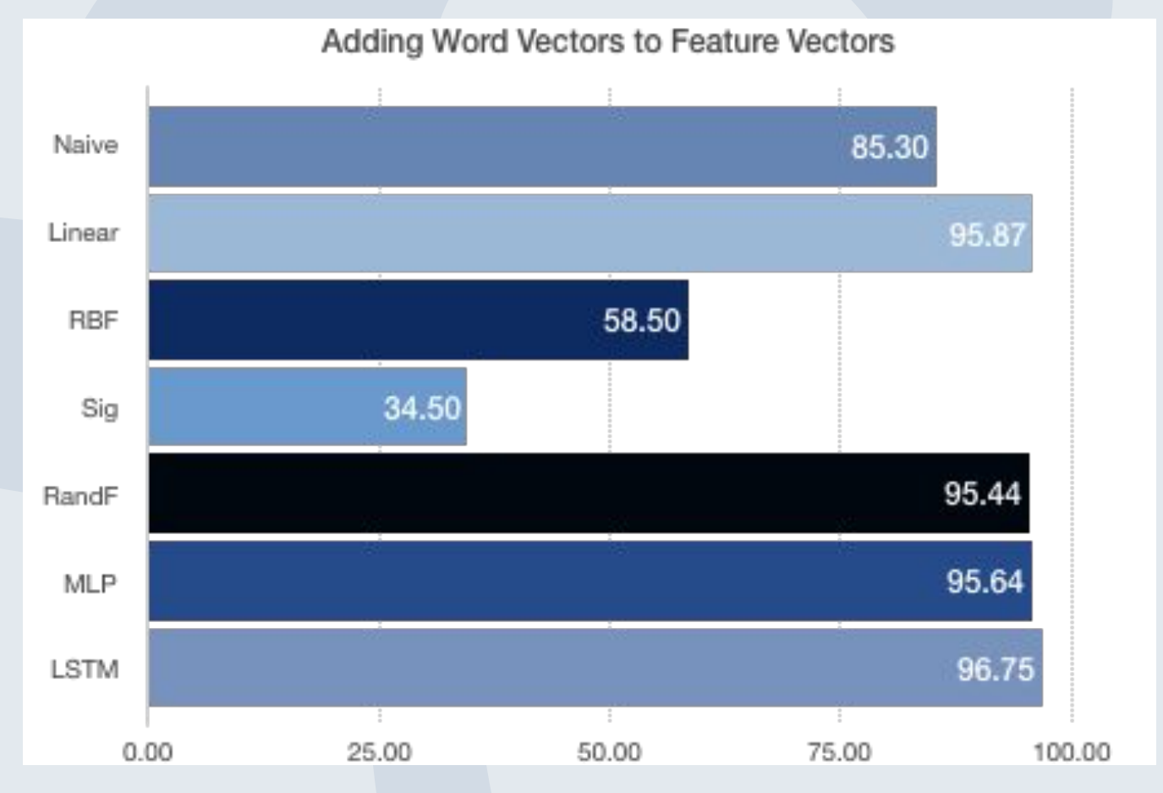
Features

FEATURE	DESCRIPTION
WORD	The word itself
WORD_LENGTH	The length of the word
IS_FIRST	True when the word is the first in a sentence
IS_LAST	True when the word is the last in a sentence
PREFIX_1	The first letter in a word
PREFIX_2	The first two letters in a word
PREFIX_3	The first three letters in a word
SUFFIX_1	The last letter in a word
SUFFIX_2	The last two letters in a word
SUFFIX_3	The last three letters in a word
PREV_WORD	The previous word before the current word
P_PREV_WORD	The previous word before the word's prev_word
PREV_TAG	The previous tag before the current word
P_PREV_TAG	The previous tag of the p_prev_word
NEXT_WORD	The word following the current word in the sentence
N_NEXT_WORD	The word following the current word's next_word in the sentence

Experimental Design

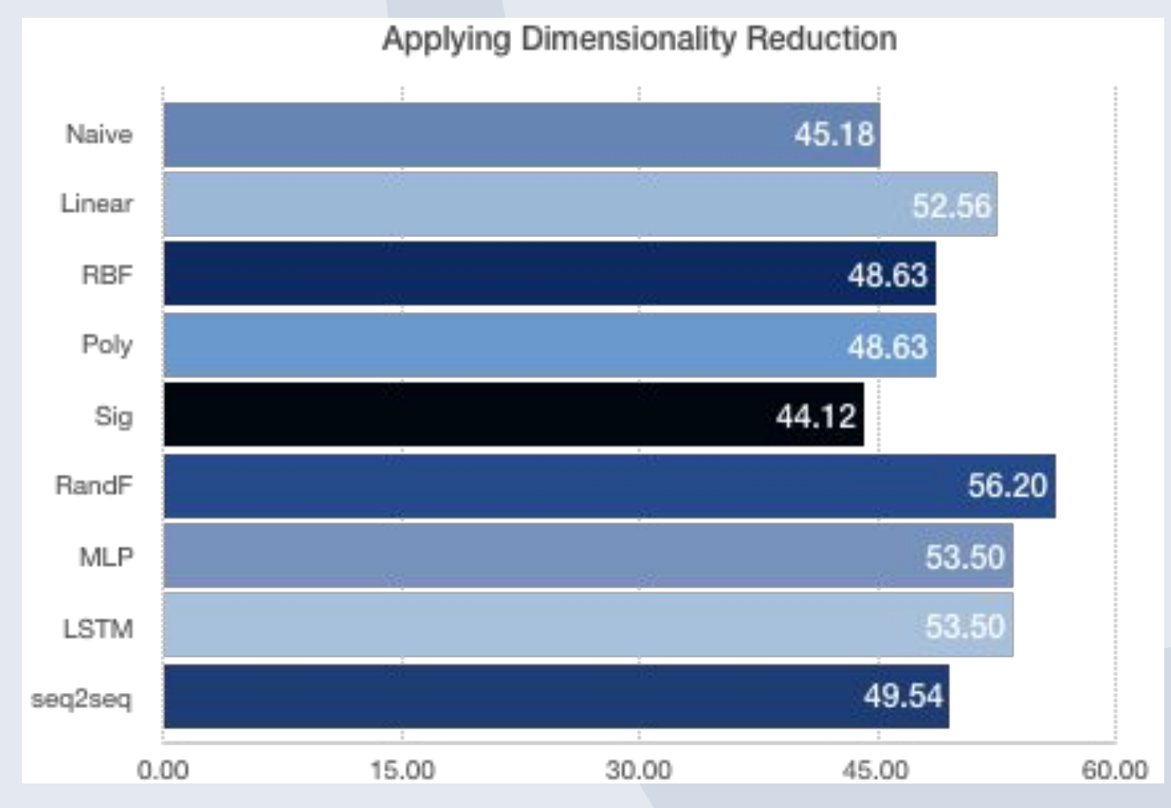
Baseline Tagger

PARTICIPANT	ACCURACY
NAIVE	85.50
LINEAR	95.72
RBF	58.50
POLY	94.67
SIGMOID	48.66
RANDOMF	92.68
MLP	91.36
LSTM	94.42



Adding Word Vector to Feature Vector

Instead of neglecting the constructed features altogether, and using only word vectors, word vectors replaced every word feature in the constructed features. This approach will result in the reduction of Baseline feature vectors from 58,200 to 9,370 dimensions.

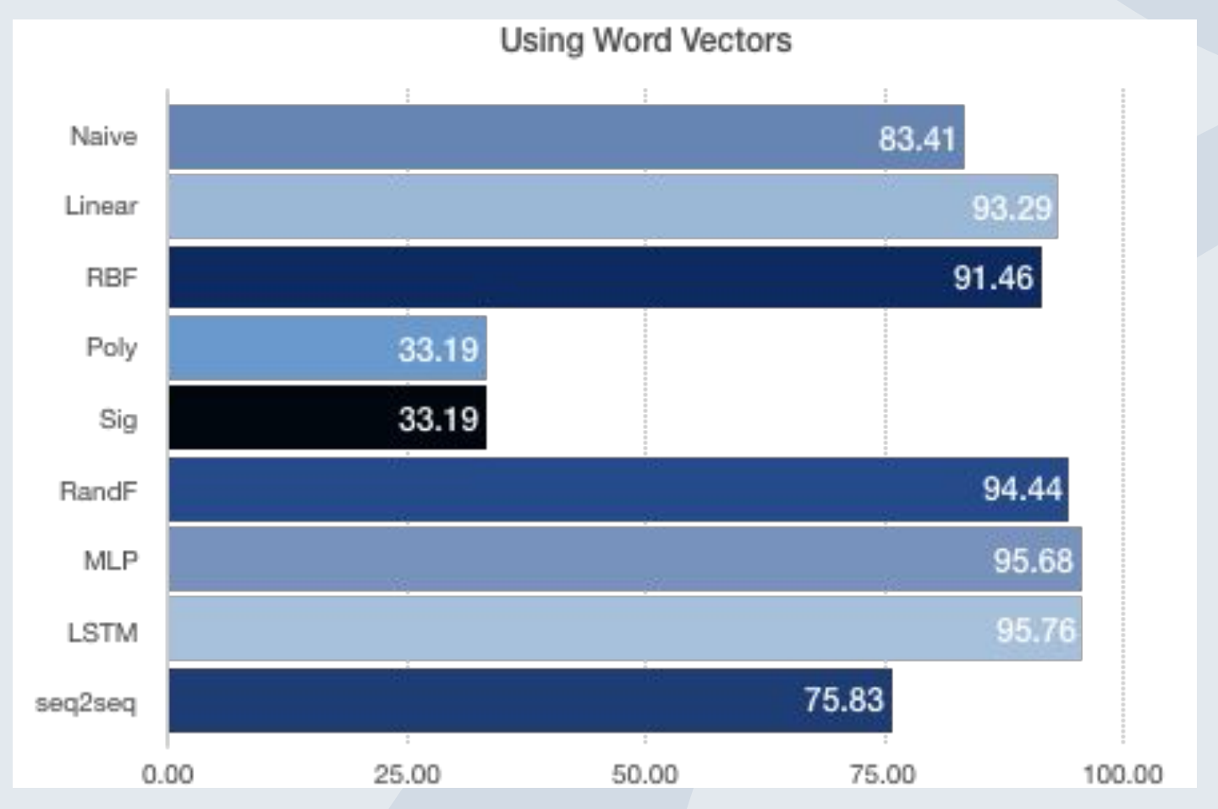


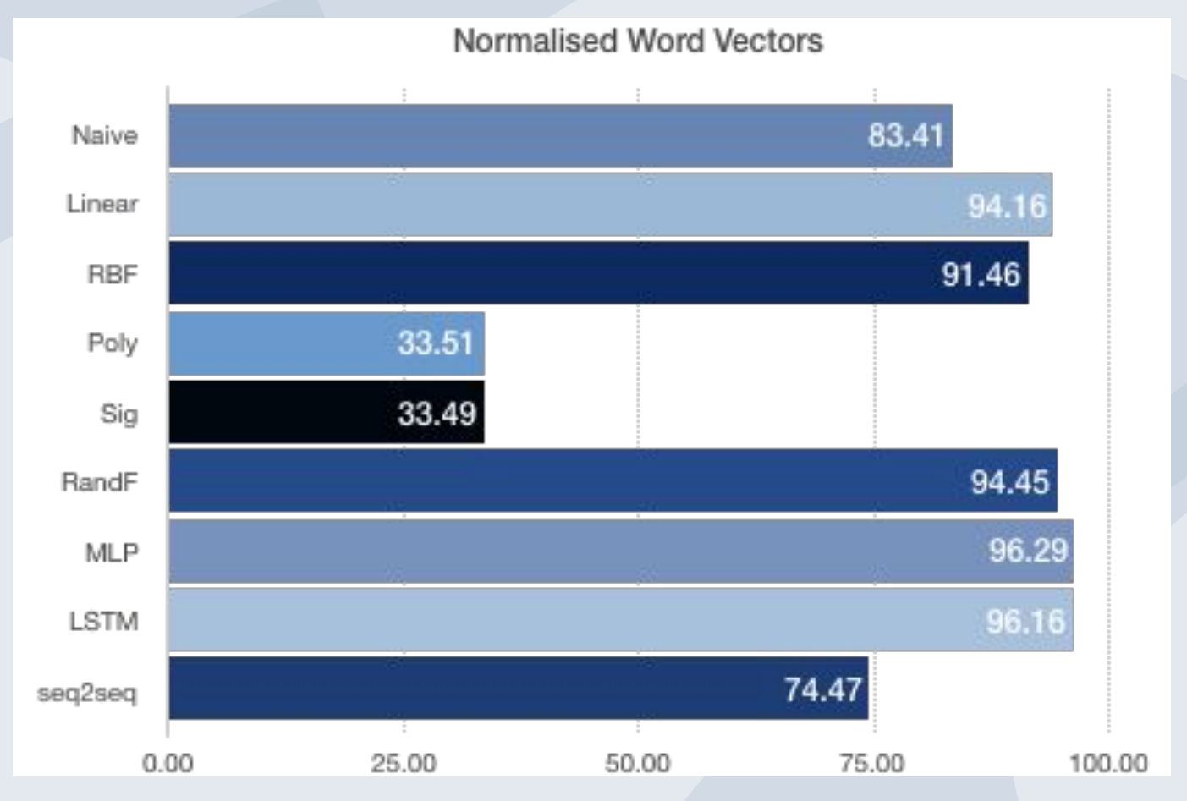
Applying Dimensionality Reduction

Since the Baseline Tagger's feature vectors are of great dimensionality, around fifty thousand. This resulted in large processing time and memory inefficiency, thus Dimensionality Reduction was applied and the feature vectors were reduced from 58,200 dimensions to only 500.

Using Word Vectors

Applying Dimensionality Reduction on the Baseline Tagger did not produce desired result, thus another approach was taken to reduce the input vectors. In this approach, the feature vectors were ignored altogether and word vectors of only 300 dimensions were used, enabling classification based on the semantic relation of words only, with no prior feature extraction.





Results and Discussion

From experimenting with different input representations, it is observed that it affected the accuracy of classifiers, either increasing the performance or deceasing it. The Table shows each tagger and the experiment that made the tagger achieve it's highest accuracy.

It can be observed from the table, that the overall highest performing classifier is the LSTM with an accuracy of 96.75%, when adding word vectors instead of constructed feature vector. With MLP classifier coming up close with an accuracy of 96.23% when using Normalized word vectors instead of feature Vectors.

CLASSIFIER	ACCURACY (%)	EXPERIMENT
NAIVE BAYES	85.50	Baseline Tagger
LINEAR SVM	95.87	Adding Word Vectors to Feature Vectors
RBF SVM	91.46	Using Word Vectors
SIGMOID SVM	48.66	Baseline Tagger
POLY SVM	94.46	Baseline Tagger
RANDOM FOREST	95.44	Adding Word Vectors to Feature Vectors
MLP	96.23	Using Normalised Word Vectors
LSTM	96.75	Adding Word Vectors to Feature Vectors
SEQ2SEQ LSTM	75.83	Using Word Vectors

Conclusion

The aim of this thesis was to build a supervised Arabic Part of Speech Tagger, with state of the Art results, experimenting with various machine learning classifiers to find the best performing classifier. It is concluded that the overall highest performing classifier, in all the carried out experiments, is the bidirectional LSTM model with an accuracy of 96.75%

Future Work

Use Transformer Word embedding models, instead of word2vec model as it doesn't take into consideration context. Thus, same word with different meanings are mapped to the same vectors (not a very good option). Such as BERT and ELMo.