Semantic Word Clustering from Large Arabic Text

The field of Arabic information retrieval has recently gained considerable attention especially with the dependency on text mining, data mining and semantic web techniques and with the rapid increase of text volume on the web where the textual data becomes high-dimensional (thousands of thousands of words in each domain) and carry semantic information.

This raise the need for word clustering techniques that can clusters words that are semantically related into meaningful groups based on their similarity.

Traditional text classification and clustering algorithms do not consider semantic relationships between Arabic words leading to inaccurately construct clusters of semantically related words.

Proposed approach

The approach consists of several stages starting from collecting text data and pre-processing, creating the classification features by averaging word vectors for all words in the text, creating the classification models, training the classification models with labelled and unlabeled data and finally creating word clusters. The stages of the approach are illustrated on Figure (1).

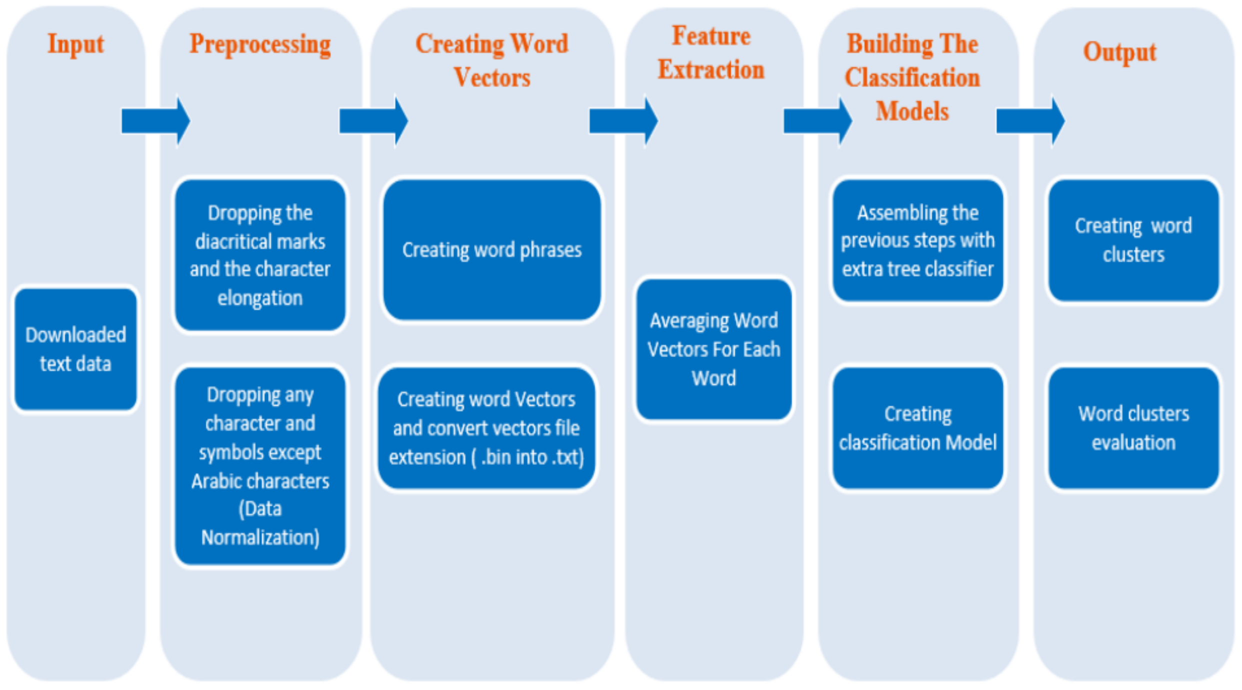

Input
Downloaded text data
Preprocessing
Dropping the diacritical marks and the character elongation
Dropping any character and symbols except Arabic characters (Data Normalization)
Creating Word Vectors
Creating word phrases
Creating word Vectors and convert vectors file extension (.bin into .txt)
Feature Extraction
Averaging Word Vectors For Each Word
Building The Classification Models
Assembling the previous steps with extra tree classifier
Creating classification Model
Output
Creating word clusters
Word clusters evaluation

Figure (1): The proposed approach

Downloading and Preprocessing Data

I download the Wikimedia database dump of Arabic Wikipedia on May 20, 2017 (https://archive.org/details/arwiki-20170520) . The volume of Arabic Wikimedia articles has reached as of September 28, 2017 above 510,651 articles (from different domains such as politics, economy, comedy, history and others). The text volume about (1.7GB) and become (1.3GB) after pre-processing. The preprocessing stage consists of dropping the diacritical marks and the character elongation and data normalization (dropping any character and symbols except Arabic characters).

Creating Word Vectors

We create word vectors using w2v model. Two stages are applied to create word vectors. First; word phrases are created to constitute the bigram statistics for the word frequency in text corpus, then they are used as better input for w2v model. Second; creating the vector representation of words using the CBOW model.

Feature Extraction

Building the classification features for the classification tasks is performed by constructing Term Frequency Inverse Document Matrix (TF-IDF). This matrix scores importance of words or terms in a document based on how frequently they appear across multiple documents. In order to construct TF-IDF matrix from the generated word vectors, we average the word vectors for each word. We used a GitHub repository class (MeanEmbeddingVectorizer) written by Nadbordrozd (Nadbordrozd, 2016)

Building the Classification models

In order to classify the generated word vectors and create the classification model, we used the Sklearn's pipeline. The pipeline sequentially applies a list of transforms (such as extracting text documents and tokenizes them) before passing the resulting features along to classifier algorithms. We used the pipeline object to transform the process of averaging word vectors and creating classification features and passes these features to extra tree classifier

Model Evaluation

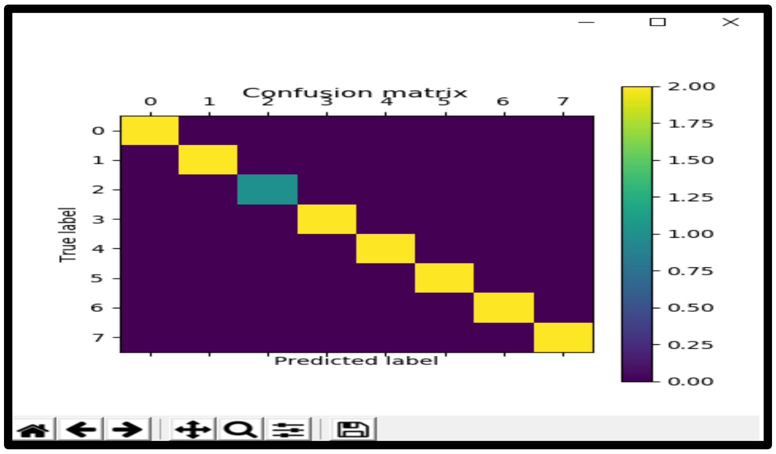
This table (1) shows the accuracy, precision, recall and f-measure results of the generated classification model with using different vector sizes, different volumes of training data and different k-folds values.

Table (1): Summary of all experiments

			251MB	414MB	643MB	781MB
Evaluation Metrics						
3 K-Fold splitting	100-word training labelled data	Accuracy	0.69	0.73	0.85	**0.88**
		Precision	0.79	0.77	0.90	**0.92**
		Recall	0.70	0.73	0.85	**0.88**
		F-measure	0.67	0.71	0.84	**0.88**
7 K-Fold splitting	300-word training labelled data	Accuracy	0.62	0.66	0.66	0.71
		Precision	0.65	0.71	0.71	0.71
		Recall	0.62	0.67	0.67	0.71
		F-measure	0.60	0.64	0.64	0.69
7 K-Fold splitting	600-word training labelled data	Accuracy	0.73	0.79	0.82	**0.85**
		Precision	0.76	0.78	0.84	**0.87**
		Recall	0.73	0.79	0.82	**0.86**
		F-measure	0.71	0.76	0.81	**0.85**
10 K-Fold splitting	1000-word training labelled data	Accuracy	0.64	0.71	0.72	0.76
		Precision	0.66	0.70	0.72	0.78
		Recall	0.66	0.71	0.72	0.76
		F-measure	0.62	0.69	0.70	0.75
15-word testing unlabeled data		Accuracy	0.73	0.93	0.93	**1.0**
		Precision	0.66	0.93	0.93	**1.0**
		Recall	0.77	0.95	0.95	**1.0**
		F-measure	0.68	0.95	0.95	**1.0**

Also, several observations can be made based on these measure values. First, the high recall and precision indicates that the models are trained well and are not under fitted. A model with high recall and low precision returns many results, but most of them are incorrected when compared to the training labels. Also, a model with high precision and low recall return few results, which are correct when compared to the training data. Second, the vector size (781MB) gives around 88% scores with 100 and 600 training labelled data, while it gives around 75% scores with 300 and 1000 training labelled data. This happens due to the insufficient vocabularies in that vector leading to less classification features that are used in the classification models. Third, the increase in both the model scores and the word vectors indicate that the bigger vector size (large and balanced data set) the better classification results. Last observation, the vector sizes 414 MB, 643MB and 781MB gives 95% and 100% scores for testing new data (unlabeled data). This means that the classification models are not over fitted. Also, this high score indicates the high-quality of the word vectors that are created.

We evaluate the last experiment (15-word testing unlabelled data) scores manually. The 15 words are: المشتري _ الزهرة_ تايوان _ سلطنة عمان_ النمسا _ سانت مارتن _ سينت مارتن_ راديوم_ كوريوم _ القولون _ مناعة_ تجاذب _ الذبذبة_ بحر العرب _بحر مرمرة. For larger testing data (unlabelled data), it would be difficult to calculate the scores manually. So, for that purpose and since our experiments are multi-class classification, the confusion matrix is an excellent method to illustrate the results of multi-class problem. Figure (2) shows the printed confusion matrix for 15-word testing unlabelled data.



As we see from Figure (2), the elements of the diagonal are the number of correct predictions and the elements off the diagonal are incorrect predictions.

In summary of these results; creating high quality word vectors requires very large plain text and balanced text, so it can be used as features with classification tasks. The ascending scores of our experiments assures the effectiveness of our approach to create high quality word clusters.