The Islamic University of Gaza
Deanship of Research and Postgraduate
Faculty of Information Technology
Master of Information Technology

الجامعـــــــة الإســـلاميـة بغـــزة
عمادة البحث العلمي والدراسات العليا
كـليـــــــــــة تكنولوجيا المعلومات
ماجستير تكـــنولوجيا المـــعلومـــات

Semantic Word Clustering from Large Arabic Text

العنقدة الدلالية لكلمات النص العربي الكبير

By

Tareq Issa Abufayad

Supervised by

Dr. Eng. Rebhi S. Baraka

Associate Professor of Computer Science

**A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Information Technology**

June / 2018

إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Semantic Word Clustering from Large Arabic Text

العنقدة الدلالية لكلمات النص العربي الكبير

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وأن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل الاخرين لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

Declaration

I understand the nature of plagiarism, and I am aware of the University's policy on this.

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted by others elsewhere for any other degree or qualification.

Student's name:	طارق عيسى أبو فياض	اسم الطالب:
Signature:	طارق عيسى أبو فياض	التوقيع:
Date:	06/06/2018	التاريخ:

بسم الله الرحمن الرحيم

الجامعة الإسلامية غزة
The Islamic University of Gaza
عمادة البحث العلمي والدراسات العليا
هاتف داخلي: 1150

الرقم: ج س غ/35/
Ref:
التاريخ: 2018/06/06م
Date:

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **طارق عيسى جبريل ابوفياض** لنيل درجة الماجستير في كلية **تكنولوجيا المعلومات**/ قسم **تكنولوجيا المعلومات** وموضوعها:

العنقدة الدلالية لكلمات النص العربي الكبير

Semantic Word Clustering from Large Arabic Text

وبعد المناقشة التي تمت اليوم **الاربعاء** 22 **رمضان** 1439هـ الموافق 2018/06/06م الساعة **الثانية عشرة ظهراً**، في قاعة **اجتماعات الكلية** اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. ربحي سليمان بركة	مشرفاً ورئيساً	.....
د. راوية فوزي عوض الله	مناقشاً داخلياً	.....
د. يوسف نبيل أبو شعبان	مناقشاً خارجياً	.....

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية تكنولوجيا المعلومات/قسم تكنولوجيا المعلومات**.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله تعالى ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

عميد البحث العلمي والدراسات العليا

أ. د. مازن إسماعيل هنية

+97082644400 +97082644800 public@iugaza.edu.ps www.iugaza.edu.ps iugaza iugaza mediaiug iugaza
P.O Box 108, Rimal,Gaza,Palestine ص.ب 108 الرمال . غزة . فلسطين

التاريخ: 20/8/07/17	الرقم العام للنسخة	3106609	اللغة	E

الموضوع/ استلام النسخة الإلكترونية لرسالة علمية

قامت إدارة المكتبات بالجامعة الإسلامية باستلام **النسخة الإلكترونية** من رسالة

الطالب/ طارق عبد الجليل أبو شاويش

رقم جامعي: 120140245 قسم: تكنولوجيا المعلومات كلية: تكنولوجيا المعلومات

وتم الاطلاع عليها، ومطابقتها بالنسخة الورقية للرسالة نفسها، ضمن المحددات المبينة أدناه:

- تم إجراء جميع التعديلات التي طلبتها لجنة المناقشة.
- تم توقيع المشرف/المشرفين على النسخة الورقية لاعتمادها كنسخة معدلة ونهائية.
- تم وضع ختم "عمادة الدراسات العليا" على النسخة الورقية لاعتماد توقيع المشرف/المشرفين.
- وجود جميع فصول الرسالة مجمَّعة في ملف (WORD) وآخر (PDF).
- وجود فهرس الرسالة، والملخصين باللغتين العربية والإنجليزية بملفات منفصلة (PDF +WORD)
- تطابق النص في كل صفحة ورقية مع النص في كل صفحة تقابلها في الصفحات الإلكترونية.
- تطابق التنسيق في جميع الصفحات (نوع وحجم الخط) بين النسخة الورقية والإلكترونية.

ملاحظة: ستقوم إدارة المكتبات بنشر هذه الرسالة كاملة بصيغة (PDF) على موقع المكتبة الإلكتروني.

130

والله ولي التوفيق،

توقيع الطالب ..... طارق أبو شاويش

إدارة المكتبة المركزية ..... محمد الراعي

Abstract

With the rapid increase of text volume on the web, 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 cluster words into meaningful groups based on their similarity, which can be used in various information retrieval tasks, like question answering systems, search engines, classification algorithms and search query expansion. In this thesis, we use word2vec model to build Arabic vector representations of words that brings extra semantic features to help building clusters of semantically related words. This involves text pre-processing, creating word vectors using word2vec model, generating the classification model and creating word clusters using Pipeline method and Extra tree classifier. We have taken the transformed text, the term frequency matrix and learn to classify the vectors with Extra Tree classifier, then we classifying and predicting the training data into the pre-defined categories. We have implemented the model and performed a set of experiments. The experiments results show the effectiveness of the model to create word clusters from a large plain Arabic text. The classification results show that the extracted features from the word vectors have empowered the classification models and achieved accuracy, precision, recall and F-measure with higher than 85%. The results also indicate that the classification model is not being under fitting (i.e., the model does not perform poorly on the training data) and it is also not being over fitting (i.e., the model performs well on both; the training data and the testing data).

Keywords: Semantic relation, Arabic Word clustering, Word2vec model, Wikipedia, Ontology

الملخص

مع الزيادة السريعة في حجم النص على الويب حيث أصبحت البيانات النصية عالية الأبعاد (الآلاف من آلاف الكلمات في كل مجال) وتحمل معلومات دلا لية. هذة الزيادة تطلبت إلى تقنيات تجميع الكلمات التي يمكنها أن تجمع الكلمات إلي مجموعات ذات معني وعلي أساس تشابهها، والتي يمكن أستخدامها في العديد من مهام أسترجاع المعلومات في محركات البحث وخورازميات التصنيف وتوسيع أستعلام البحث. في هذة الرسالة نقترح أستخدام أداة أو نموذج "word2vec" لبناء المتجة التمثيلي لكلمات النص العربي الكبير والتي سوف تعطي معاني ومميزات دلالية للمساعدة في بناء مجموعات دلالية من كلمات النص العربي الكبير. وهذا يتضمن المعالجة المسبقة للنص، بناء المتجة التمثيلي بأستخدام نموذج "word2vec"، بناء نموذج التصنيف والمجموعات الدلالية بإستخدام طريقة "Pipeline" وخورازمية التصنيف "Extra tree classifier". تم قمنا بأخذ النص الذي تم معالجتة ومصفوفة تردد المصطلحات لبناء مصنف المتجهات بإستخدام خورازمية التصنيف ""Extra tree classifier" وأستخدامة في تصنيف وتنبؤ الكلمات إلي الفئات المحددة مسبقا. قمنا بتطبيق نموذج التصنيف وإجراء تجارب عديدة بإستخدام النموذج، حيث أن النتائج أظهرت إلى فعالية النموذج لإنشاء مجموعات دلالية من النص العربي الكبير. وتظهر نتائج التصنيف إلي أن السمات المستخرجة من كلمات المتجهات قد مكنت نموذج التصنيف من تحقيق دقة عالية وصحة بأكثر من 85%. كما أن النتائج تشير إلي أن نموذج التصنيف لايخضع إلي حالة "under fitting" (أى أن النموذج لايؤدي أداء ضعيفا علي بيانات التدريب)، وأيضا لايخضع إلي حالة "over fitting" (أى أن النموذج يؤدي أداءً جيداً علي كل من بيانات التدريب والإختبار).

الكلمات المفتاحية : علاقات دلالية، عنقدة الكلمات العربية، أداة Word2vec، الويكبيديا، الانتولوجيا

Dedication

Praise be to Allah SWT for His blessings, which are countless, O God, make us thankful for your blessings.

Thankful to Almighty Allah SWT who gave me the strength and patient to finish this work. And prayers and peace be upon my great teacher and messenger, Muhammed (May Allah SWT bless and grant him), who taught us the purpose of life, and his family and companions and followers and those followed them in charity until the Day of Judgment.

My dedicated also goes to

My beautiful parents …...who never stop giving me their supports in all times

My dearest wife…...who lives with me moment by moment the life affairs and helps me through these obstacles with light of hope and support

My twin children…...Issa & Safa…...My Allah SWT blessing on me

My beloved brothers and sisters…...who stands with me all the time without boring and hustle.

To …...those who care about me.

Acknowledgment

In the Name of Allah, the Most Merciful, the Most Compassionate all praise be to Allah, the Lord of the worlds; and prayers and peace be upon Mohamed His servant and messenger.

First of all, Praise to Allah SWT for giving me the health and strength to complete this thesis.

I am thankful for my supervisor Dr. Rebhi S. Baraka, without his help, guidance, and continuous follow-up; this research would never have been done.

Also I would like to thanks the academic staff of the Faculty of Information Technology at the Islamic University-Gaza who helped me during my Master's study and taught me different courses.

Last but not least, I am greatly grateful to my family for continued love and support.

Tariq Issa Abufayad
June, 2018

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CBOW	Continuous Bag-of-Words
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
LSI	Latent Semantic Analysis Indexing
LSA	Latent Semantic Analysis
TF	Term-Frequency
TF-IDF	Term Frequency, Inverse Document Matrix
TP	True Positive
TN	True Negative
FP	False Positive
FN	True Positive
NLP	Natural Language Processing
SG	Skip-Gram
SVD	Singular Value Decomposition
VPS	Virtual Private Server
WV	Word Vectors
W2V	Word2vec

Chapter 1
Introduction

The field of Arabic information retrieval has recently gained considerable attention especially with the dependency on text mining, data mining and semantic web techniques (Al-Zoghby, Ahmed, & Hamza, 2013). This gain is due to 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 (Alotaibi & Anderson, 2017).

Traditional text classification and clustering algorithms do not consider semantic relationships between Arabic words leading to inaccurately construct clusters of semantically related words. Many researchers have used external resources to overcome this problem. Alkoffash (2012) has employed two clustering algorithm, k-means and k-medoids on Arabic text to extract a feature set of keywords in order to improve the clusters quality. Bloehdorn, Cimiano, and Hotho (2006) have used integrated conceptual features extracted from ontologies to improve text clustering and classification. Hotho, Staab, and Stumme (2003) have used conceptual terms from WordNet to improve text clustering and resulting more semantically related clusters.

Thus there is a need for semantic word clustering to create clusters of semantically related words. These clusters can be used in various information retrieval tasks, like question answering systems, search engines, classification algorithms and search query expansion (Baker & McCallum, 1998).

The aim of our thesis is to build semantic word clusters from Arabic large plain text. These clusters can be used by the Arabic semantic applications. To the best of our knowledge, our thesis presents a new way to extract semantic word clusters from large Arabic plain text. Existing approaches like Arabic WordNet contain just concepts extracted from Wikipedia without considering the semantic meanings between words.

We use word2vec (w2v) model (Mikolov & Dean, 2013) to build Arabic vector representation of words that brings extra semantic features to help building clusters of semantically related words. There are models for creating vector representation of words. The two most popular ones are word2vec and glove (Akata, Reed, Walter, Lee, & Schiele, 2015).

w2v is a two-layer neural network is trained to predict a set of targets words from a set of context words. It has two main models or architecture for the target prediction: skip-gram (SG) and continues bag-of-words (CBOW). In SG, the model uses the center word to predict the surrounding words. In CBOW, the model uses a window (number of words) of word to predict the middle of word. w2v is a memory friendly than glove.

Glove is another predictive model created by Pennington, Socher, and Manning (2014). Glove works like w2v, except that the glove input is about matrix that holds the ratio of the co-occurrence probabilities of two words, while w2v takes the entire corpus as input.

After creating the vector representation of words, we take the mean or the average of all vectors corresponding to individual words to construct the term frequency matrix for the classification process. Then, we take the transformed text (the processed text) and the term frequency matrix as input for Extra Trees classifier algorithm. Finally, using the classification model to classify and predict the testing data into their class. Next we state the problem of the research, the derived objectives. Then we indicate the important of this work followed by scope and limitation of the research. We then describe the methodology we follow in our research. Finally, we give the organization of the thesis.

1.1 Statement of the problem

As the growth of text data on the web increases, it becomes high-dimensional (thousands of thousands of words in each domain) and carry semantic information. Therefore, there is a need for word clustering techniques that can clusters words into meaningful groups based on their similarity. Traditional text classification and clustering algorithms do not consider the semantic relationships between Arabic words so they cannot accurately construct clusters of semantically related words.

The problem of this thesis is how to build an Arabic semantic word clustering from large text taking into consideration the semantic relationships between words.

1.2 Objectives

1.2.1 Main objective

The main objective of this research is to create groups of Arabic semantic word clusters from large Arabic plain text. These clusters should have high intra-cluster similarity (words within a cluster are similar) and low inter-cluster similarity (words from different clusters are dissimilar).

1.2.2 Specific objectives

The specific objectives of the research are:

1) To collect large Arabic text from the web, such as Arabic Wikipedia XML dump file, or any other free Arabic corpus founded online.
2) To perform pre-processing on the collected text such as, removing Arabic diacritics (tashkil and Tatweel), removing extra spaces between words and none Arabic characters
3) To build Arabic vector representation of words using a couple of models including glove and w2v.
4) To create the term frequency matrix for the generated vectors by averaging the word vectors for each word.
5) To pipeline generated features with the transformed text data to learn to classify the word vectors with Extra Trees classifier and creating the classification model.
6) To train the classification model with new training data and predicting the testing data into their pre-defined categories. There are different class labels (word clusters) created and each class has its unique words.
7) To evaluate the generated clusters using most accepted and used metrics in text mining field such as precision, recall, f-measure and confusion matrix.

1.3 Importance of the research

The importance of the research stems from creating Arabic vector representation of words to create high similarity clusters of semantically related words. These clusters can be used by the Arabic semantic applications like question answering systems, search engines and query expansion.

The generated Arabic vector representation of words is updatable vector and not a static vector. This feature increases the scalability of the generated clusters by allowing to enhance the vector with new text and generate new Arabic vector characterized by new sematic related words this provides continuous availability resources for various Arabic applications in information retrieval and semantic applications.

1.4 Scope and limitations

1. The Arabic vector representation of words is generated using w2v models.
2. The implementation of w2v is found in a couple of programming languages such as Java, C and Python. We use Python due to its efficiency and widely uses in the data science projects.
3. w2v has two architecture options which are SG (default) or CBOW. We have experimented with CBOW architecture since it is faster in training.
4. Experiments conducted only on Arabic text since it is our chosen domain.
5. For results evaluation, we use the following metrics: confusion matrix, precision, recall and f-measure which are the most appropriate and proved effective in such situations.

1.5 Methodology

The methodology used to achieve the research objectives comprises the following steps:

1. **Collect Arabic Texts**
   We download the Wikimedia database dump of Arabic Wikipedia on May 20, 2017. The text volume is about 1.7GB and it is a collection of written Arabic articles in various domains.

2. **Pre-Processing**

The downloaded text contains various noisy symbols and characters such as question marks, dashes, under scores, extra spaces, numbers, dollar sign, diacritical marks and character elongation. We perform two processes to normalize the text to be used as a better input for the next step. First process: dropping the diacritical marks and the character elongation (tatweel) using Pyarabic Python library (Zerrouki, 2010). Second process: remove all noisy symbols and keep only the Arabic letters using our Python script which is effective for large text (see Appendix (A.1) for more information).

3. **Build the Vectors Representation of Words**

We build the vector representation of words using w2v model. It accepts the input as sequence of words as one-hot encoding which is not a better input (see Section (2.1) for more information). Also w2v accepts the input as a phrase which is a better input for creating high quality word vectors. We use the word2phrase algorithm to join adjacent pairs of words that appear at least five times in text with an '_' character.

4. **Generate the Classification Features**

To describe the frequency of words in the text corpus we build the term-frequency (TF) matrix from the generated word vectors. We use a GitHub repository class (Mean Embedding Vectorizer) written by Nadbordrozd which averages word vectors for all words in a text (Nadbordrozd, 2016). We construct TF matrix to create the classification features for the next classification process.

5. **Generate the Classification Model**

To generate the classification model we must assemble the transformed text data, the generated classification features and the extra tree classifier to classify word vectors. To implement this, we use the Sklearn's pipeline library which assembles several steps that can be cross-validated together while setting different parameters.

6. **Generate Word Clusters**
   To generate word clusters from text corpus, we fit the generated classification model with training (labelled and unlabelled data) and testing data. The model classifies the testing words into their predefined categories (class labels) and each class label (category) has its unique words.

7. **Evaluate Results**
   To evaluate the generated clusters, we use the most common metrics in this field such as confusion matrix, precision, recall and f-measure. They are explained in Section (4.6.2).

The rest of the thesis is organized as follows: Chapter 2 describes the theoretical and technical foundation of our approach. Chapter 3 reviews related works. Chapter 4 describes the proposed approach. Chapter 5 presents the experiments and the results. Finally, Chapter 6 presents the conclusions and future works.

Chapter 2
Theoretical and Technical Foundation

This chapter presents an overview of the theoretical aspects alongside with the technical foundation used to create the vector representation of words. It gives an overview about word clustering techniques, the different models that create word vector, the software environment, libraries and tools for creating word clusters, the issues and complexity in creating word clusters from large unstructured Arabic text, and finally the evaluation and the quality of a generated word clusters.

2.1 The Complexity of Creating Word Clusters from Large Unstructured Arabic Text.

Creating word clusters from large unstructured Arabic text is a complex process. It requires to takes a lot of considerations before cleaning the text and it is considered a trade-off process due to the following reasons:

- The richness of the morphological and the grammatical nature of the Arabic language leads to create very large and growing word vectors. This is due the different forms a word can take, for example if we added suffix (ا) to the verb (ذهب) it becomes (ذهبا) which means past simple event and also means that it's done by two persons (Duwairi, 2006).

- If we consider to use the stemming approach and reducing the words to their root pattern, the text words can be reduced from thousands to hundreds (Eldos, 2003). Also, this leads to lose the semantic meanings between the words and creates unbalanced text and thus poor word vector quality.

- The problem of the lack of Arabic NLP tools and resource that benefit from the capabilities provided by semantic web technologies such as intelligent reasoning over data, semantic search and data interoperability (Al-Khalifa & Al-Wabil, 2007).

In summary of these challenges we can say that creating word clusters require a technique that can be able to preserve the semantic relations between words and at the same time keeps the text balanced and this is what we try to achieve in this research.

2.2 Overview of Word Clustering Techniques

Many applications in Natural Language Processing (NLP) benefit from the use of word clustering or document clustering, it improves the performance on many NLP tasks such as information extraction, information retrieval, search engine and machine translation (Denkowski, 2009). The task of word clustering is considered as unsupervised classification technique and can be viewed as text clustering problem. Each cluster has its unique words and a context that is different from other clusters. To define the similarity between these words or data points in high-dimensional space a distance measure is used such as Euclidean, Cosine, Jaccard, and Edit Distance. Clustering algorithms fall under three primary categories:

- **Partitional Clustering**

  Partitional clustering algorithms divide the data points or objects into non-overlapping clusters(subsets) such that each point is in exactly one subset and the data points within a cluster are similar. The algorithms require the number of clusters to be generated in advance, which can be used in many applications like defining the static routes for network performance analysis. One of the most used **p**artitioning algorithms is k-means clustering; in which, each cluster is represented by the center or means of the data points belonging to the cluster and is sensitive to anomalous data points and outliers. K-medoids clustering; in which, each cluster is represented by one of the objects in the cluster. k-medoids is less sensitive to outliers compared to k-means. Clara algorithm is an extension to k-medoids for large data sets. Figure (2.1) illustrates the concept of partitional clustering, where the data points are non-overlapping.

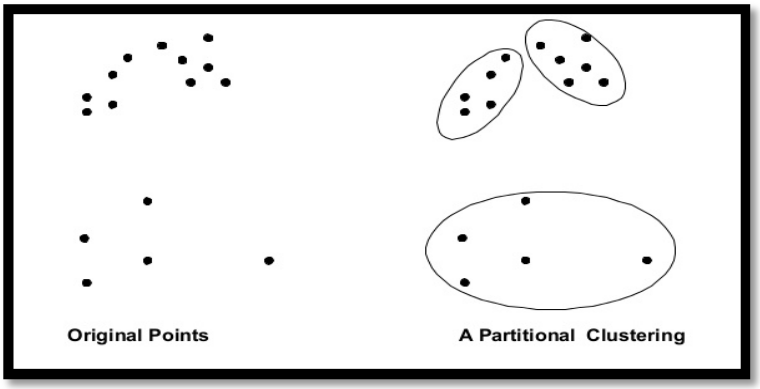

Figure (2.1): Example of partitional clustering (slideplayer, 2018)

- **Density Based Clustering**

  Density based clustering algorithms do not require the number of clusters in advance (as parameter). Rather they infer the number of clusters based on the density approximation of data. It creates clusters of any shape in a dataset containing noise and outliers. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is the most well-known density based algorithm. It works efficiently on large databases with no prior knowledge of the number of clusters required. The idea of DBSCAN is that for each point of a cluster, the neighborhood of given radius has to contain at least minimum number of points. Figure (2.2) illustrate the clusters which are the dense region in the data space, separated by regions of lower density points.

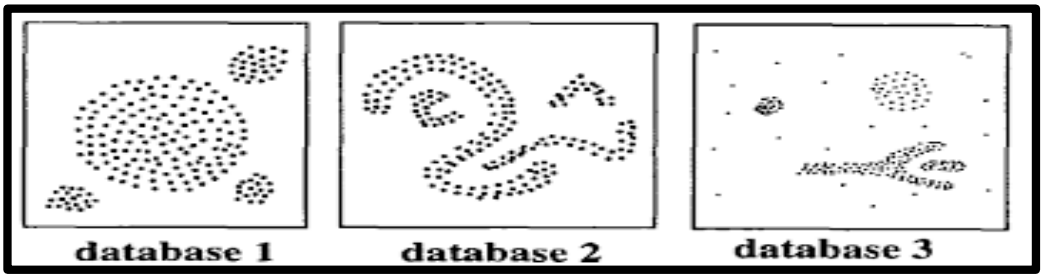

Figure (2.2): Example of density-based clustering
(Ester, Kriegel, Sander, & Xu, 1996)

- **Hierarchical Clustering**

Hierarchical clustering algorithms divide the data points into sets of nested clusters that are organized as a tree. There are two types of this method: Agglomerative; which start with the points as individual clusters, and at each step, merge the closest pair of clusters, and divisive method; which start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. The distance between each cluster is measured with three different methods. Method one: Single Linkage, the distance between two clusters is defined as the shortest distance between two points in each cluster. Method two: Complete Linkage, the distance between two clusters is defined as the longest distance between two points in each cluster. Method three: Average Linkage, the distance between two clusters is defined as the average distance between each point in each cluster to every point in the other clusters. Figure (2.3) illustrate an example of hierarchical clustering, which clusters group of countries (25 country) into three different clusters according to human development index in listed countries.

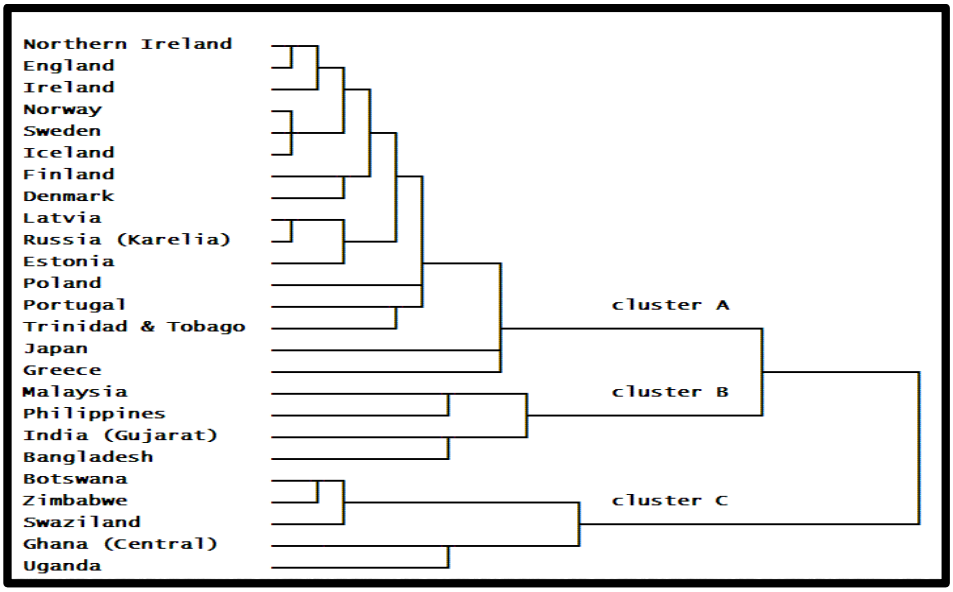

Figure (2.3): Example of hierarchical clustering method
(Asia Pacific Forum, 2005)

2.3 Vector Representation of Words

2.3.1 Definition of Vector Representation of Words

Vector representation of words is a way to represent a word or document as a vector. In traditional vector space models, a word is represented by one-bit position in huge vector (called one hot encoding). For example, if we have a vocabulary of 10 words, and "Palestine" is the 4th word in the text, it would be represented by: 000100...00. This representation treats word as atomic and does not provide meaningful comparison between words other than equality. Also, this representation results in word vectors that are extremely sparse.

2.3.2 The Need of Using Word Vector

Word Vectors (WV) provide a fresh perspective to most problems in NLP tasks. It becomes an alternative to the classical methods that solve these problems. For example, WV does help with identifies the similar words and synonyms in large text. On the other hand, using the classical methods (edit distance, WordNet, stemming and using dictionaries) to identify word similarities, requires a lot of human efforts and more than one tools.

WV are used with various NLP applications such as, machine translation, part-of-speech and named entity recognition, relation extraction, co-reference resolution, clustering, semantic analysis of documents and sentiment analysis (Maas et al., 2011).

2.4 The Different Models that Create Word Vectors

Many of WV models are implemented to create the vector space representation of words; they have different ratio of capturing the semantic relatedness between words. These vector models either are predictive models (also known as neural word embedding models) or count based model (also known as distributional semantic models). Predictive models learn their vectors by trying to predict the neighbor's words. While count models or distributional semantic models learn their vectors by computing the statistics of how often some word co-occurs with its neighbor words in a large text corpus, and then map these counts-statistics to vector for each word (Baroni, Dinu, & Kruszewski, 2014). We

just concentrate on predictive models which are relate to our work since they are computationally efficient than count models for learning word embedding from raw text. There are two famous predictive models that creates word vectors from corpus, which are: Word2vec model and Glove model. Next, we elaborate in each of them.

2.4.1 Word2vec Model

w2v model, developed by Mikolov and Dean (2013), has gained a lot of attention in the recent years. w2v is a feed-forward neural network that learns in unsupervised way the representations of the word vectors to capture the semantic relationships. Also, w2v is a shallow learning algorithm which is computationally less expensive than other learning models. w2v uses two architectures for learning and creating word vectors, which are: SG and CBOW.

- **Continuous bag-of-words (CBOW)**

The CBOW model predicts target words (e.g. sat) from source context words (The cat sat on floor) It assumes that there is only one word considered per context, which means the model will predict one target word given one context word, which is ***like*** a bigram model (Rong, 2014). CBOW model uses a window (number of words) of word to predict the middle of word. Figure (2.4) depicts an example window word of size two to predict the target word (sat) from the source context words (The cat sat on floor).

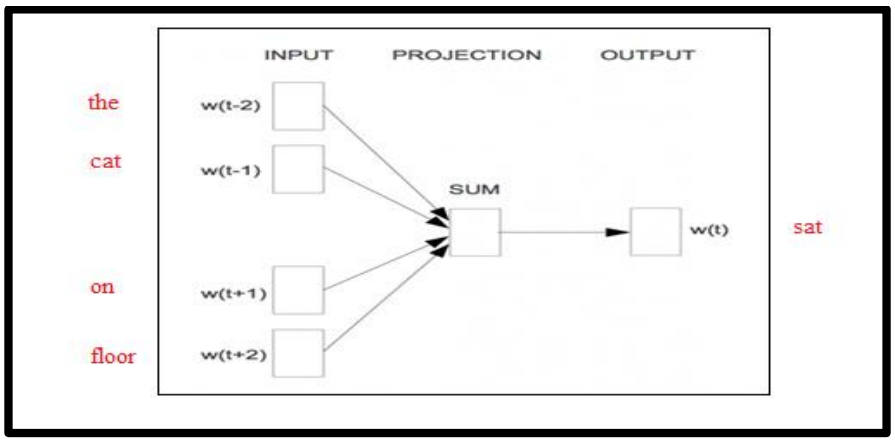

Figure (2.4): Continuous bag-of-words model (Mikolov & Dean, 2013)

To calculate the probability of the next word given the previous words, CBOW uses objective function to train the neural network and compute the target words probabilities by summing the log probabilities of the source words as shown in Equation (2.1)

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} log\ p(w_t \mid w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n}) \qquad \textbf{(2.1)}$$

The objective function J_θ receives a window of n words around the target word (wt) at each time step (t) and calculates the probability of the target word. CBOW is several times faster to train than the skip-gram, slightly better accuracy for the frequent words. Also, it uses continuous representations whose order is of no importance.

- **Skip-gram (SG)**

SG model uses the center word to predict the surrounding words in window as shown in Figure (2.5).

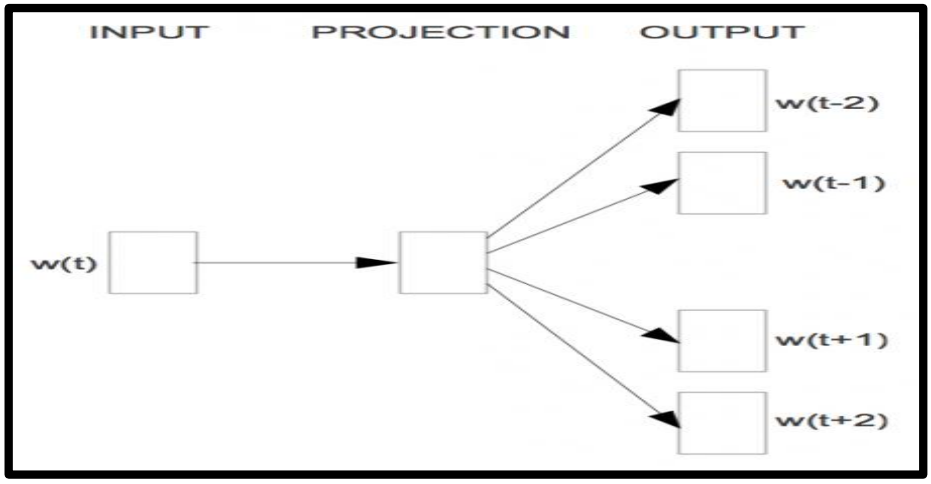

Figure (2.5): Skip-gram model (Mikolov & Dean, 2013)

The SG objective function (Equation 2.2) sums the log probabilities of the surrounding n words to the left and to the right of the target word (wt).

$$J_\theta = \frac{1}{\mathrm{T}}\sum_{t=1}^{T} \sum_{-n \le j \le n, \neq 0} \log p\left(w_{t+j} \mid w_t\right) \qquad \textbf{(2.2)}$$

SG works well with small amount of training data and represents well even rare words or phrases.

2.4.2 Glove Model

Glove is another predictive model created by Pennington et al. (2014) . Glove works like w2v, except that the glove input is a matrix that holds the ratio of the co-occurrence probabilities of two words, while w2v takes the entire corpus as input. The glove algorithm consists of the following steps:

- Collect word co-occurrence statistics in a form of word co-occurrence matrix X Each element X_{ij} of such matrix represents how often word i appears in context of word j.

- Define soft constraints for each word pair as stated in Equation (2.3)

$$w_i^T w_j + bi + bj = \log X_{ij} \quad \textbf{(2.3)}$$

  w_i is the vector for the main word, w_j is the vector for the context word, b_i and b_j are scalar biases for the main and context words.

- Define a cost function as stated in Equation (2.4)

$$J = \sum_{i,j=1}^{v} f(X_{ij})\left(w_i^T w_j + bi + bj - \log X_{ij}\right)^2 \quad \textbf{(2.4)}$$

  Wherew_i , w_j , b_i, and x_j are parameters to learn, v is the size of the vocabulary. $f(x)$ is a weighting function which the performance of the model depends on it.

Next, the software environment, libraries, as well as tools used to develop our approach to create word clusters are represented.

2.5 Software Environment, Libraries and Tools for Creating Word Clusters

Creating Arabic word clusters from large text is a challenging task. In order to create these clusters, different tools are needed besides our main approach. These tools are used to generate the Arabic word clusters from large Arabic plain text.

2.5.1 The Software Environment

Performing text pre-processing on large text files consumes a lot of computer RAM, time consuming process and sometimes cause computer halt and therefore lose a data. Also creating word vectors from large text requires a lot of computer RAM. We performed these processes using rented virtual private server (VPS). We used another environment for creating the word clusters since it requires less computer resources (see Section 5.2 for more information).

2.5.2 The Downloaded Arabic Text and the Pre-processing Tools

We downloaded the Wikimedia database dump of the Arabic Wikipedia on May 20, 2017 (Wikipedia, 2017). The corpora are collection of articles written in various domains such as politics, art, religion, health, economic, etc. The articles contain a lot of none Arabic characters like English and Spanish. Also, they contain the diacritical marks, extra spaces, dashes, question marks, dollar signs, character elongation, etc.

We have performed text pre-processing in two stages. Stage one, removes the diacritical marks and character elongation, e.g., (فلســـــطين). We use an external Python library written by (Zerrouki, 2010). Stage two, normalize, the text by keeping only Arabic characters. We write a Python script to remove none Arabic characters and symbols (see Section 4.2 for more information).

After cleaning and normalizing the downloaded text, we create word phrases from the text. These word phrases constitute a better input for w2v model and thus create high quality word vectors. Word2phrase is a w2v package tool that uses bigram statistics to form phrases in text corpus based on a minimum and maximum frequency.

2.5.3 Python Libraries

We use two Python libraries to help in creating Arabic word clusters. First, we use a couple of Skylearn Python libraries such as Extra tree classifier algorithm to classify the word vectors. Pipeline library to assemble several steps that can be cross validated together while setting different parameter. K-Fold library which splits training data into training and testing data and metrics library to calculate the model score accuracy, precision, recall and confusion matrix metrics. Second, we use gensim library to convert the generated word vector model file (.bin) to (.txt) file for later use in the classification process, see section 4.2 for more information.

In summary of these steps, we created these categories "clusters" from the training text and each cluster has its own unique vocabulary. In addition, we can add new cluster label "classification label" and make the prediction again.

2.6 The Evaluation of Generated Word Clustering

We evaluate the classification results using the most used and accepted metrics in text classification field. We use the confusion matrix to validate performance of the generated classification models, since it is considering an excellent evaluation metrics of multi-class classification problem. To improve reading of the classification results, the confusion matrix results can be plotted, since the elements of the diagonal are correct predictions, and far elements are incorrect predictions. Table (4.1) describes the used terms of the confusion matrix, where the columns are the predicted class, and the rows are the actual class (Kohavi & Provost, 1998).

Table (2.1): Confusion matrix illustration

	Predicted: No	**Predicted: Yes**
Actual: No	True Negative (TN)	False Positive (FP)
Actual: Yes	False Negative (FN)	True Positive (TP)

- **True Negative (TN):** refer to the number of correct prediction that an instance is negative
- **False Positive (FP):** refer to the number of incorrect prediction that an instance is positive
- **False Negative (FN):** refer to the number of incorrect prediction that an instance is negative
- **True Positive (TP):** refer to the number of correct prediction that an instance is positive.

Also, we use another classification evaluation metrics such as accuracy (Equation 2.5), precision (Equation 2.6), recall (Equation 2.7) and F-measure (Equation 2.8) which are the most accepted metrics in data mining tasks.

- **Accuracy:** the total number of predictions that were corrected. It is determined using the following equation.

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN} \quad \textbf{(2.4)}$$

- **Precision:** the total number of the positive predicted that were corrected. It is determined using the following equation.

$$Precision = \frac{TP}{TP + FP} \quad \textbf{(2.5)}$$

- **Recall:** the total number of positive predictions that were correctly identified. It is determined using the following equation.

$$Recall = \frac{TP}{TP + FN} \quad \textbf{(2.6)}$$

- **F-measure:** it is another standard measure used to measure the performance of the classification models. This measure comes handy when the total number of

negative cases greater than the number of positive cases. It is mean parameter and calculated using the precision and recall evaluation metrics.

$$F - measure = 2\frac{Precision * Recall}{Precision + Recall} \quad \textbf{(2.7)}$$

2.7 Summary

In this chapter, we have presented an overview of existing clustering algorithms and how they used the distance function to define the similarity between words in a dimensional space. Also we give a theoretical overview about the proposed approach, we illustrate the definition of word vectors and how they overcome the difficulty of encoding representation technique and the use of word vectors in common NLP tasks. The most famous models that used to create the vector representation of words are; word2vec model and glove. We have explained the environment used to implement the different parts of the proposed approach, then we have described the downloaded corpus and the tools that are used to pre-process the downloaded text, then we introduced the Python libraries that are used to create the word clusters. After that, we have explained the problems of creating word clusters from large unstructured Arabic plain text, and now the proposed approach can help to overcome these problems. Finally, we have presented evaluating methods for the generated word clusters by using the most accepted and used matrices in this field, which are: precision, recall, f-measure, accuracy and the confusion matrix.

Chapter 3
Related Works

This chapter presents a review of researches that are related to our research and identifies their drawbacks, limitations and draw conclusion. In order to present related works better, we categorize them into three groups: group one, summarizes the traditional text classification and clustering techniques for large text. Group two, summarizes word vectors based classifying and clustering techniques. Group three, summarizes other models for creating word vectors.

3.1 Current Text Classification and Clustering Techniques for Large Text

Many of existing text mining algorithms use the bag-of-words model, where text terms are used as a feature for representing the document. This model ignores the semantic relationship among words (Heap, Bain, Wobcke, Krzywicki, & Schmeidl, 2017). Recent researches have improved the classification and the clustering process by enhancing the feature selection process and integrating other methods alongside classification and clustering process. The related works in this section is fallen under into two subsections; section one, related works that works on feature selection enhancing. Section two, related works that uses external methods and developed techniques for large text classification and clustering process.

3.1.1 Feature Selection Enhancing

Bloehdorn et al. (2006), improved the classification and the clustering tasks by integrating conceptual features that are constructed automatically from ontologies. They used OHSUMED (Hersh, Buckley, Leone, & Hickam, 1994) which is a medical text collection and consist of 54,708 documents. Although their approach shows a competitive result and overcomes the limitation of bag-of-words model, it still faces the following problem: it captures just the concepts that are labelled by noun phrases from the ontology, this generates less semantic concepts and less rich features.

Al Tarouti and Kalita (2016) used the words embedding technique to enhance and automatically construct the WordNet for low-resource language and they experimented with Arabic language. They used the w2v model for generating the vector representation for Arabic words using three freely available corpora; watan-2004 corpus (12 million words), Khaleej-2004 corpus (3 million words) and the Wikipedia Arabic articles (21 million words). They used the generated word vectors to enhance the translation method for automatic WordNet construction. Their method produced Arabic synonymous with 78.4% precision and semantically related words up to 90.4% precision. Despite the high precision value for the semantically related words, the generated word vectors still are not of high-quality vectors because they don't do the pre-processing steps for the collected words and they depend on changing the w2v settings like window size to obtain high precision. The text pre-processing is necessary to generate a balanced vector.

Nabil, Atiya, and Aly (2015) combined linguistic methods with statistical methods to generate Arabic key phrases that provide semantic meaning as they can be used in many NLP applications. They developed a stemmer and part of speech tagger and trained on Arabic tree bank corpus which consists of about one million words in various topics. Then they used the TF-IDF as statistical method to measure the similarity candidate patterns generated by their POS tagger and the document titles. Also, they used different statistical methods like w2v model. In summary, their research concludes that using statistical features in the key phrase extract task leads to better results if corpus is well annotated. Their work is different from ours, but both of us used w2v model to create the semantic vectors with different quality. Also, they didn't implement a standalone text pre-processing tasks on the used text before creating the vectors, they just depended on the POS tagger and stemmer pre-processing methods which produced unbalanced text and hence unbalanced vectors be generated and will affect the system performance.

3.1.2 External Methods and Developed Techniques

Abu Tair and Baraka (2013) developed a parallel classifier for large Arabic text classification task. The work based on the K-Nearest Neighbour (K-NN) Algorithm which is known as one of the best classification algorithms, but requires a large amount of computational power for high dimensional text. To overcome this, they implemented the proposed classifier in parallel using the Message Passing Interface (MPI) on multicomputer cluster. The proposed classifier achieved accuracy, precision, recall and f-measure around 95%. Also, they have used the Bag-of-Words (BOW) document indexing to represent the text documents. In addition to that they had performed a couple of pre-processing tasks on the corpus like string tokenization, stop words removal, stemming and light stemming to enhance text document representation process and create more semantic feature vector. Despite the approach achieves high results, using stemming and light stemming delete much semantic information between words and thus the feature vector does not capture the complete semantic meanings between words.

Huang (2008) used different similarity measures with partitional clustering algorithm for processing large datasets. The approach relies on the BOW techniques to represent the documents and their contents. It applies on the following measures: cosine similarity, Jacquard correlation coefficient, Pearson correlation coefficient, Euclidean distance and Kullback Leibler divergence in order to find the closeness distance between a pair of terms or objects and thus create accurate clusters which are semantically related. The last measure gives slightly better results in their clustering results and creates more balanced and closer clusters. The Jacquard and Pearson coefficient create more coherent clusters but not semantically related. Additionally, the quality of the generated clusters can be affected by the following factors: the representation of the documents or the feature selection, as we mentioned before that BOW representation suffers from ignoring the semantic meaning between the words. Also, the different results of the similarity measures indicate that these measures can affect the quality of the generated clusters. The cluster algorithm itself can affect the results, as in partitional clustering it is fast but requires to determine the number of clusters before creating the clusters, while hierarchical clustering algorithm is slower and gives good results for categorical data (Xu & Wunsch, 2005).

Al-Shalabi and Obeidat (2008) improved the Arabic text classification task with using K-NN classifier and N-Grams document indexing instead of BOW indexing. They used online Arabic corpus collected by Mesleh (Moh'd A Mesleh, 2007) and it consists of 1445 documents that vary in length and classified into nine categories. The average accuracy in case of using N-Gram is 0.7353 while in single term indexing is 0.6688. Despite using both word-level unigrams and bigrams in document indexing, the average accuracy is slightly different than that using BOW indexing. Their approach still does not capture the semantic information among the document words and less semantic features are created.

3.2 Current Researches and Approaches that Use Word Vectors in Classification and Clustering of Large Text

Ma and Zhang (2015) used w2v to process big data. They aimed to select useful features or decrease the feature dimension from processing big data. They trained the w2v model with 20 Newsgroups dataset to create WV, then they applied the linear calculation for each word vector, and found semantic related words, after that, they grouped words together using k-means. They aimed from this strategy to decrease the data dimension and speed up the multi-class classification process and thus the new dimension will decrease the time cost in the machine learning tasks. Both of our approach and this approach benefit of creating word vectors. We use the word vectors to create the classification features, while their approach used the word vectors to decrease the data dimension by selecting new feature dimension. Their approach gives accuracy score for the classification performance on average of five different data dimension 77%. While our approach attempts to give classification model accuracy of 85% on average. Their work is time consuming to create similar words as they applied linear calculation for each word vector then clustered these words using k-means algorithm, while our approach attempts to average word vectors and use this as a feature for the classification process.

Baker and McCallum (1998) used the distributional clustering technique to cluster words into groups. By using distributional clustering, they reduced the feature dimensionality by three orders of magnitude and lose only 2% accuracy on 20 Newsgroups dataset which contains about 20,000 articles. They created clusters of words based on the distribution of

the class labels associated with each word, then they use feature reduction for text classification problem. In our approach, we create word clusters based on the semantic similarity between words, while their approach considered the distributional theory which sometimes creates misleading distribution and less quality.

Wei et al. (2015) proposed a semantic approach for text clustering that overcomes the traditional clustering algorithm problem (does not consider the sematic relationships among words). They added semantic information from ontology such as WordNet which is widely used to improve the text clustering task. They built a lexical chain to extract core semantic features that expressed the topic of documents, the lexical chain is restricted only for four types of relations: identity, synonymy, hypernym, and meronym. To find the similarity between concepts, they proposed a modified similarity measure based on WordNet for word sense disambiguation. In WordNet, concepts are nodes and semantic relations between these concepts can treated as edges. They compute the similarity between two concepts by finding the least common node that connects these concepts, then the distance between two concepts is computed. However, WordNet lexical databases not including all semantic relationships between words. Also, WordNet focuses on verbs, adjectives and nouns which leads to less semantic concepts.

Wu (2014) presented a theoretic clustering algorithm to improve the process of phrase recognition and achieved 95% model accuracy. To achieve that, he designed a top-down division clustering algorithm based on the information gain theory, clustered similar words from unlabelled texts and used them as features for the classification process. He used iterative k-means clustering algorithm to create word clusters. Our work is similar to Wu work except in two points. First point: his work is directed to English language while ours is directed to Arabic language and it can be generalized to any other language and is not affected by the language characteristics and grammar. Second point: we created word vectors through w2v model which is the most famous and used in current NLP researches, while Wu created word vectors by using iterative k-means clustering algorithm with support of other contextual information like POS tag information.

Alotaibi and Anderson (2017) have applied word clustering technique to improve the sentiment analysis task. They applied the brown clustering algorithm to create word clusters. It is a hierarchical clustering algorithm which generate clusters of words depending of their context within the same data set, as similar words have similar distribution of words to their left and right. They experimented with Arabic language and developed their corpus from four different genres; news, reviews, user market reviews, and movie reviews. After creating word clusters, the cluster label of the word is used as a feature for the later classification process. Their research emphasized the potential gain of using word clustering as a feature for classification. Our work is almost similar to their work except in the following point: they used the brown clustering algorithm to generate word clusters where it does not consider the semantic information between words and depends on the distribution of words to group similar words, while our approach uses neural networks to create hidden layers that able to group similar words and preserve semantic information between words.

Lilleberg, Zhu, and Zhang (2015) have classified documents with using word2vec. They used skip-gram model to classify the documents. To create the features for the classification process they added weights to each word based on its frequency within the document, they create a weighted sums of word vectors to represent document combined with tf-idf. They experiment with 20 newsgroup dataset which have 18,000 newsgroup posts on 20 topics. They achieve 90% accuracy score. We adapt the same approach, excepts that we created the classification features by averaging the word vectors.

In the context of Arabic applications, to the best of our knowledge, our work presents a new way to build word clusters from large Arabic plain text, but closer applications have been introduced. Al Zamil and Al-Radaideh (2014) extracted semantic relationships from multiple datasets. These datasets represent Classical Arabic (Holy Qur'an), Modern Standard Arabic (newspapers), and unstructured Arabic texts (social blogs). To enrich the semantic relationships extraction process, they used Arabic WordNet (AWN) which helps to obtain the lexical structure of the patterns. AWN contents come from the translation process of English *synsets* to Arabic *synsets* which does not take in consideration the

complex grammar nature of the Arabic language, thus less rich semantic relationships are found in AWN.

3.3 Other models for creating word vectors

Beside the neural network models to create word vectors, there is a statistical computations models applied to corpus in order to create word vectors. These models like Latent Semantic Indexing (LSI) but most commonly referred to as Latent Semantic Analysis (LSA), Probalistic latent semantic analysis (PLSA), Latent Dirichect Allocation (LDA) are popular methods and have good research and community support (Alghamdi & Alfalqi, 2015). LSI is a linear model and it's not a best solution to handle non-linear dependencies. LSI model create sparse vector spaces for words and uses Singular Value Decomposition (SVD) to reduce vectors dimensions. SVD is mathematical method that can reduce a N-dimensional dataset into fewer dimensions, different methods exist such as Principle Components Analysis, Factor Analysis, and so on. SVD is designed for normally- distributed data but is not appropriate for count data (Rosario, 2000).

Al-Anzi and AbuZeina (2017) used the cosine similarity measure and latent semantic indexing (LSI) to enhance Arabic text classification task. They used LSI to represent the textual data as numeric vectors and maintain the semantic information between the words. LSI produced a matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity among columns. They experimented with a corpus which contains 4000 newspaper documents belonging to 10 different categories. They generated the textual features of the corpus using LSI-SVD technique and experimented with 8 classification methods, and the results showed that support vector machine and k-nearest neighbours classifiers which has the top performance compared to the other classifier. LSI is a distributional model designed for normally distributed-data, so inappropriate for count data, also SVD representation need more storage and computing tie. So it is not efficient representation compared to deep neural networks (Barbara, 2000).

3.4 Summary

In this chapter, we have presented the research's that are related to our research, and identified their drawbacks, limitations and draw conclusions. We categorize them into three groups, which are: group 1; current text classification and clustering techniques for large text, many of these researches using the BOW model. In this model a text words are used as a feature for representing the document and use these features in document classification task. However, this model does not capture the semantic information among the document words and hence less semantic features are created. Other researches overcome the limitation of BOW model by integrating conceptual features (rich semantic features) that are constructed automatically from ontologies. Other researches use N-Grams model (the number of words taken together as a single entity) with BOW model, the N-Grams are unigrams(n=1), bigrams(n=2), etc., however the model's accuracies resulted of using these word-level unigrams and bigrams in document indexing is not promising and is slightly different from using BOW indexing. Group 2; the current researches and approaches that uses word vectors in classification and clustering of large text have used the generated word vectors for different tasks such as selecting useful features, decreasing the feature dimension and improving the text clustering task. In the context of Arabic researches, to the best of our knowledge, our work presents a new way to build Arabic word clusters from large Arabic plain text, but closer application has been introduced. Al-Anzi & AbuZeina used the cosine similarity measure and latent semantic indexing to enhance Arabic text classification task. Using the latent semantic indexing with a mathematical technique called singular value decomposition, semantic features are extracted from 4000 news document for the classification task. However, the latent semantic indexing is not an efficient representation for capturing the semantic relationships between text words, as it is designed for normally distributed data. Group 3; other models for creating word vectors, these models are linear models not uses the neural network as word2vec. These models like LSI uses statistical computations calcualtions applied to corpus in order to create word vectors. However these models easy to impelement, understand and use, but not an efficeinet for skewed distribution data.

Chapter 4
Clustering Words Semantically

In this chapter, we present our approach for creating clusters of semantically related words from large Arabic plain text. 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 unlabelled data and finally creating word clusters. The stages of the approach are illustrated on Figure (4.1).

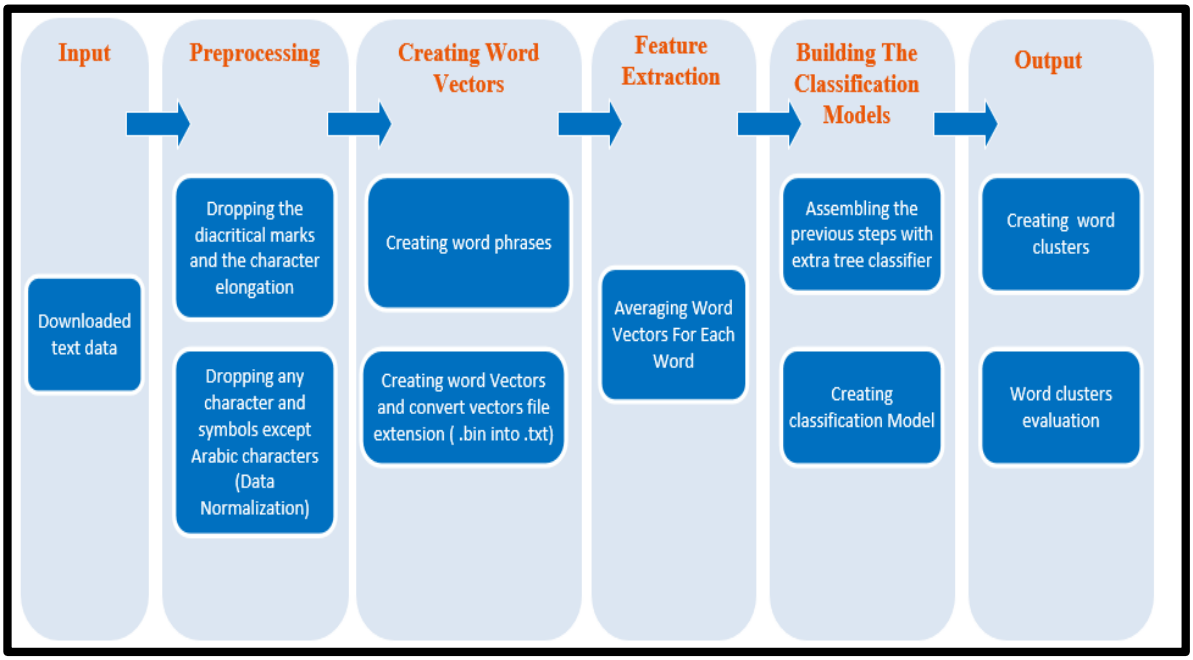

Figure (4.1): Generating the word clusters (the proposed approach)

It consists of six stages: stage 1 is concerned with collecting Wikimedia database dump of Arabic Wikipedia on May 20, 2017. Stage 2 is pre-processing the downloaded text which consists of two sub processes. Process 1 removes the diacritical marks and character elongation and process 2 normalizes the text by removing extra space, dashes, and keep only Arabic text. Stage 3 creates word vectors which consists of two sub processes.

Process 1 creates the word phrases from the pre-processed text, these phrases are used as better input for creating word vectors and process 2 uses these word phrases to create the word vectors using w2v model. Stage 4 prepares the classification input (features selection) by averaging the generated word vectors for all the words. Stage 5 builds the classification models which consist of two sub processes. Process 1 assemblies the generated classification features, the generated word vectors and process 2 implementing process 1 by using the skylearn pipeline library. Stage 6 gives the output which consists of two sub processes. process one creates the word clusters by training the classification model with labelled and unlabelled data and classify the testing data into their predefined categories and process 2 evaluates the classification models with the most used and accepted measures in text classification problem like confusion matrix, accuracy, recall, precession and F-measure (Al-Shalabi & Obeidat, 2008). Finally, we obtained hundreds of predicted words which are classified into their predefined categories. We consider these class labels the clusters which are different from each other and have similar words in each cluster (class label). We elaborate in each stage of these stages.

4.1 Collecting Text Data

Arabic language is a rich morphological language and spoken by more than 420 million people around the world and becomes the sixth most spoken language in the world (Istizada, 2017). We use the Wikimedia database dump of the Arabic Wikipedia articles to perform our experiments.

The Wikimedia database dump is a collection of Arabic Wikipedia articles written by hundreds of active Internet users (630 users around the world) and is the fifth best Wikipedia edition among other languages. 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.

We generate different volumes of Wikimedia text in order to create different sizes of word vectors that contain different vocabulary sizes which might affect the quality of the generated word clusters. These volumes are as follows:

- **304** Megabytes of Wikimedia dump
- **608** Megabytes of Wikimedia dump
- **1200** Megabytes of Wikimedia dump
- **1700** Megabytes of Wikimedia dump

4.2 Text Pre-Processing

Text pre-processing and cleaning is a trade-off process (see Section 2.5 for more information), so we need large plain text and basic cleaning methods that can preserve the semantic information between the words and hence create high quality word vectors and not losing too much text data during pre-processing. Therefore, we perform two basic steps.

4.2.1 Dropping the Diacritical Marks and Character Elongation

Many Arabic words or characters within the downloaded Wikipedia text have the diacritical marks. Although these diacritical marks enrich the Arabic words with different meaning in different context, they are still on ongoing research on integrating the diacritical marks in the semantic web application and different NLP tasks (Chennoufi & Mazroui, 2017) (Amrouche, Falek, & Teffahi, 2017). For that reason, we removed the diacritical marks to create word vectors with unique and not ambiguous vocabularies. Also, many Arabic words in the downloaded Wikipedia text have character elongation or Tatweel (like: كتـــــابي). This also affects the input for creating word vectors.

To remove the diacritical marks and strip the character elongation for large Arabic text we used an external Python library called PyArabic 0.6.2 (Zerrouki, 2010).

4.2.2 Data Normalization

The downloaded Wikipedia text contains various none Arabic characters, under scores, dashes, numbers, extra spaces and other symbols. To remove these things and keep just the Arabic characters, we wrote a Python script to normalize the text and generate plain cleaned Arabic text. We used a regular expression that search the downloaded text line by line for Arabic characters and join these characters, then we write the output of this script in new text file that has only Arabic characters.

With these pre-processing methods, we preserve the semantic meaning between words and avoid deleting useful information. Also, we generate a clean text input to create high quality word vectors and hence generate word clusters with high accuracy, precision, recall and F-measure results.

4.3 Create Vector Representation of Words

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.

4.3.1 Creating Word Phrases

For creating better input for w2v, we group up similar words (like "ضغط جوي" to "ضغط_جوي") and use word2phrase tool which uses bigram statistics to form a phrase with two words based on a minimum and maximum frequency of adjacent pair of two words. Figure (4.2) shows sample of the generated word phrases.

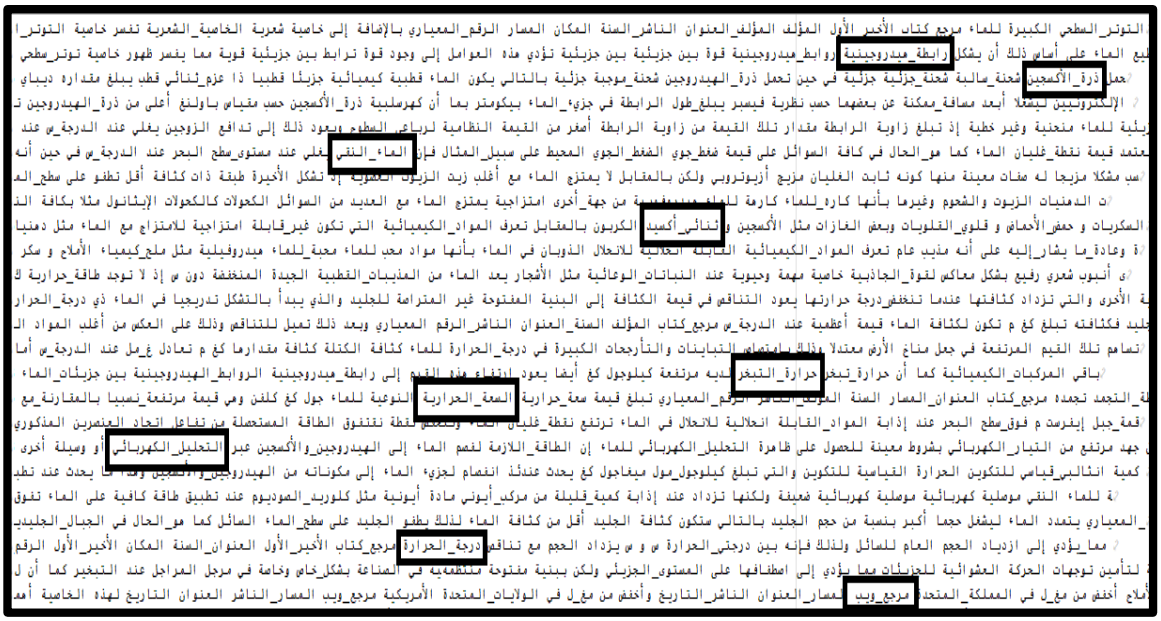

Figure (4.2): Sample of generated word phrases

4.3.2 Creating Word Vectors

Word2vec is unsupervised machine learning algorithm that does not need any human annotation to learn. The output of w2v (the generated word vectors) can be used in clustering algorithms to create clusters of semantically related words. To achieve this step, we experiment with the following pre-processed sentence as shown in Figure (4.3).

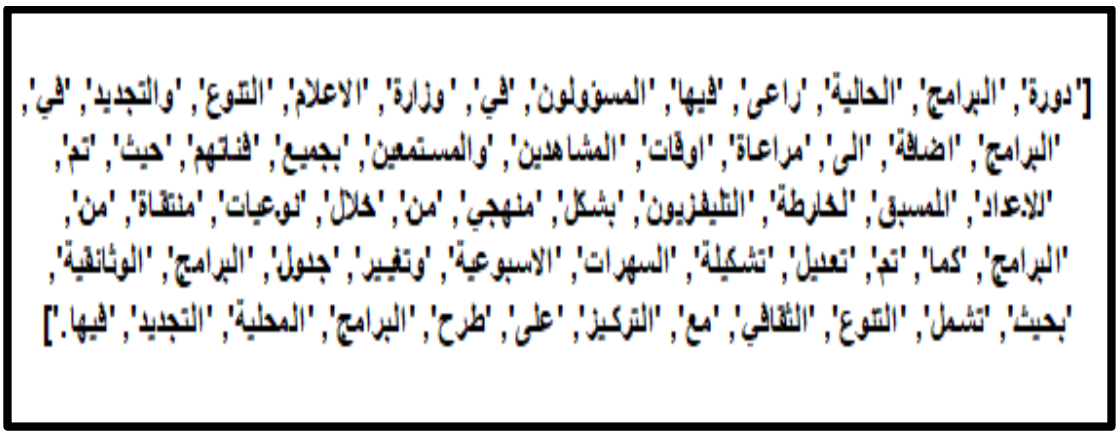

Figure (4.3): Pre-processed sentence example

After applying w2v and generating word vectors, we have applied k-means clustering algorithm to create clusters of semantically related words from the previous sentence. Figure (4.4) shows the final clusters that are generated.

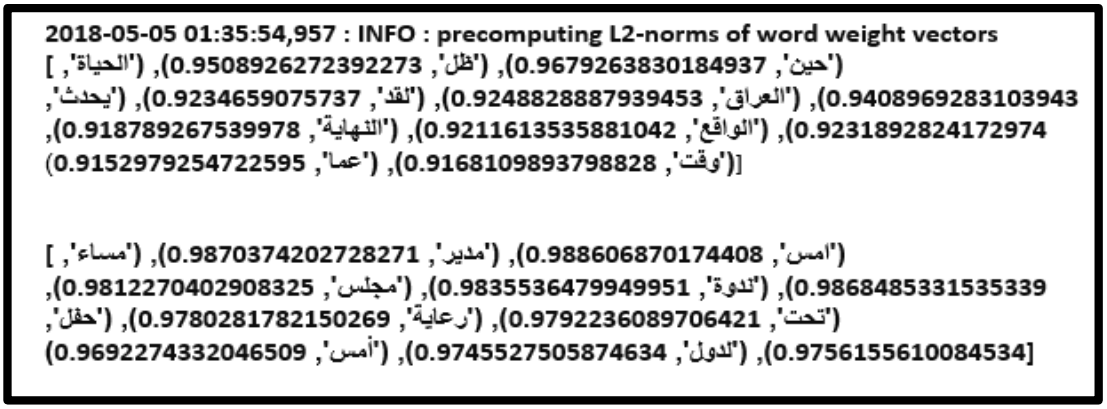

Figure (4.4): Sample of generated word clusters using w2v

From Figure (4.4), we notice that the word clusters are not semantically related and applying k-means clustering algorithm to w2v results is not promising and does not lead to effectively creating semantically related word clusters. So to further enhance our

approach and creating better results, we apply Extra Tree classification algorithm to the output of w2v. This brings extra semantic features that help in word classification and thus create groups (categories) of semantically related words. This same approach has been adapted by Lilleberg et al. (2015) in classifying document with using w2v. They used the output of wordv2ec with support vector machine. See Section (3.2) for more information.

4.4 Building the Classification Features

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). Figure (4.4) illustrates the sparse matrix of generated classification features, the number in bracket is the index of the value in the matrix (row, column) and 1 is the number of times a term appeared in document.

```
C:\Python27\python.exe C:/Users/tariq/PycharmProjects/ww2v/classification.py
  (0, 257004)    1.0
  (1, 86895)     1.0
  (2, 205207)    1.0
  (3, 77775)     1.0
  (4, 185087)    1.0
  (5, 13180)     1.0
  (6, 185024)    1.0
  (7, 91867)     1.0
  (8, 192222)    1.0
  (9, 28327)     1.0
  (10, 237859)   1.0
  (11, 192988)   1.0
  (12, 228507)   1.0
  (13, 196272)   1.0
  (14, 74771)    1.0
  (15, 187767)   1.0
  (16, 9208)     1.0
  (17, 15908)    1.0
  (18, 272222)   1.0
  (19, 72286)    1.0
  (20, 189195)   1.0
  (21, 18034)    1.0
  (22, 197790)   1.0
  (23, 19771)    1.0
  (24, 224674)   1.0
  : :
  (274164, 131822)   1.0
  (274165, 213663)   1.0
  (274166, 64801)    1.0
```

Figure (4.5): Sparse matrix of classification features

4.5 Building the Classification Model

In order to classify the generated word vectors and create the classification model, we need to chain the previous steps (the generated classification features and classification algorithm) together.

To achieve this, 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. Extra tree classifiers standing for extremely randomized trees proposed by Geurts, Ernst, and Wehenkel (2006). Extra tree classifier is an extremely randomized version of decision tree classifier and it is productive in the context of a large number of numerical features. Figure (4.5) illustrates the data flow for the pipeline process.

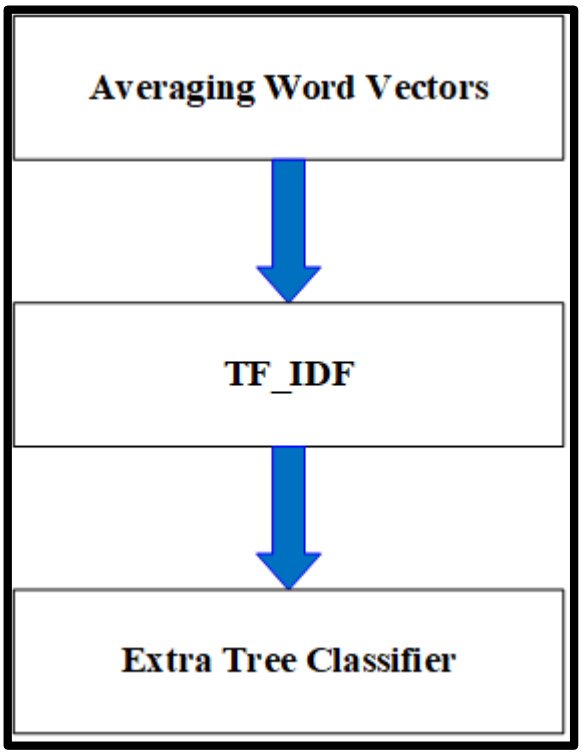

Figure (4.6): The pipeline data flow

After running the pipeline process, the generated classification model is generated and is ready to be fitted with labelled and unlabeled training data.

4.6 Output

This is the final stage in our approach to create word clusters. Before training the generated classification model, we need to split the data set into k non-overlapping subsets (folds), which generate independent training and unseen testing data. Finally, we evaluate the generated word clusters with confusion matrix, precision, recall and f-measure metrics. This stage consists of two main steps:

4.6.1 Creating Word Clusters

In order to create word clusters, we need to split the data set into independent training and unseen testing data. We use Python sklearn k-fold cross validation package. It splits dataset into k consecutive folds, each fold is then used a validation set once while k-1 remaining folds form the training set. By using k-fold cross validation, we avoid fitting the generated classification model with insufficient training data. Then we fit the classification model with different volumes of labelled and unlabelled training data in order to create different word clusters volumes.

The output of the classification model is multiple unique class with multiple similar words for each class. Therefore, we consider each class as cluster with similar words. Figure (4.7) depicts the results of one of multiple experiments that have been made to create word clusters. It shows the predicting classes for the testing words, the model's accuracy and the plotting of the confusion matrix for the classification model.

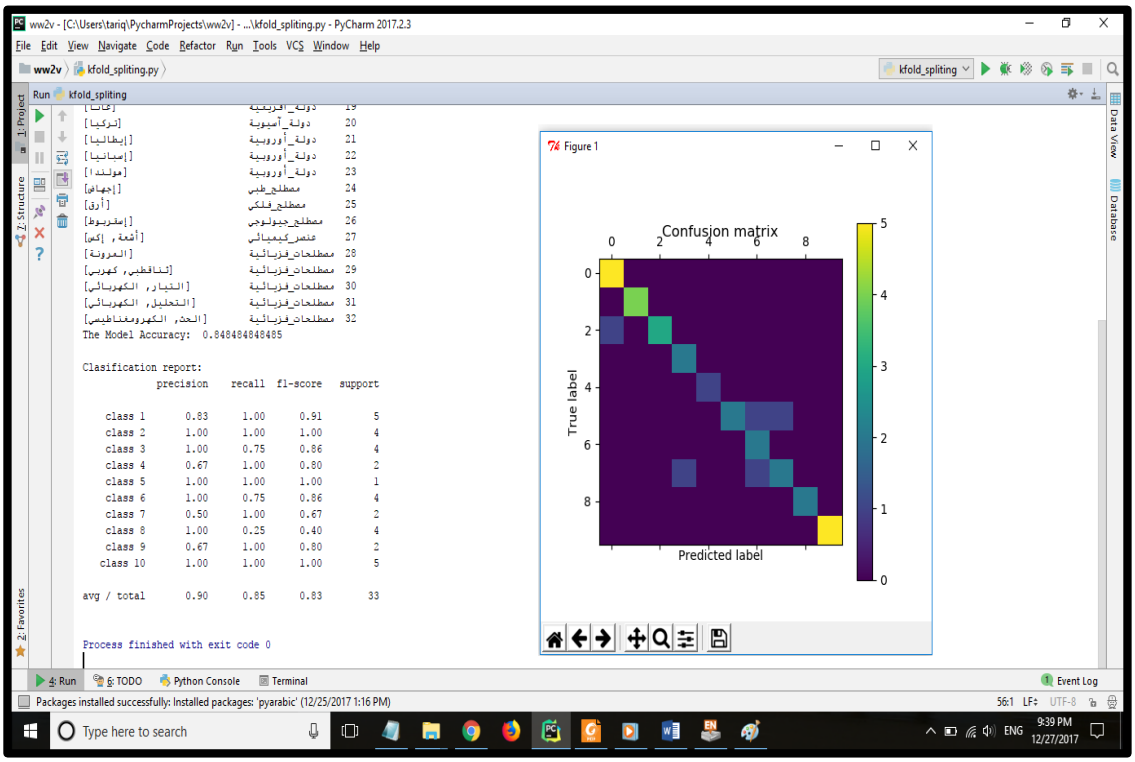

Figure (4.7): Example of generated classification results

4.6.2 Word Clusters Evaluation

In order to evaluate the quality of the generated clusters four measures are used: measuring the correctness of extracted patterns with respect to existing correct ones using a recall metric, measuring the ability of our proposed approach to detect patterns with respect to all retrieved information using a precision metric, denoting the overall accuracy by applying an f-measure metrics, and finally, constructing the confusion matrix to visualize the performance of the created classification models.

The evaluation results for the classification model gives very good percentage for predicting labelled data and gives 100% model accuracy score for unlabelled data. See Section (5.3) for the details.

4.7 Summary

We have presented the approach to generate word clusters from large Arabic plain text. To the best to our knowledge, it is a novel approach which applies the w2v to text classification to generate the word clusters.

We have created word clusters from large Arabic plain text and used the w2v model to create such word clusters, but its gives low semantically related word clusters in the favour of Arabic language and good semantically related word clusters in the favour of English language. This is due to the complex morphological nature of the Arabic language and lack of NLP tools for the Arabic language. Our approach benefits from using w2v to create the semantic word vectors and preserve the semantic information between the text words and then do the classification process to classify these semantic vectors and create the word clusters.

In chapter 5, we present the experiments we performed to evaluate the approach and discuss the results.

Chapter 5
Experimental Results and Evaluation

This chapter covers the experiment results and shows the effectiveness of our approach to create high quality word clusters (the words within a cluster are similar and dissimilar from different clusters) from large Arabic plain text. The chapter includes three sections: Section 5.1 discusses the experiment design in details. Section 5.2 discusses the results of our experiments. Section 5.3 summarizes the chapter.

5.1 Experimental Design

In this section we describe and explain the following factors related to the experiments; the corpus characteristics, the experimental environment, the experimental parameters and how do we tune these parameters, and the experimental procedures.

5.1.1 The Corpus Characteristics

We use the Wikimedia database dump of Arabic Wikipedia articles as of May 20, 2017 to perform our experiments. These articles are written by hundreds of internet active editors around the world. The articles volume is about (1.7GB) and becomes (1.3GB) after pre-processing. These articles are written for multiple domains or topics such history, health, sport, polices, and politics.

The content of Wikipedia articles is well written and reviewed. It follows the Wikipedia policy and guidelines for writing articles. The articles should be clear, be as concise as possible, using common sense which emphasize the spirit of the rule, avoiding over linking and should not contradict each other (Wikipedia, 2017).

We divided the downloaded Wikipedia text volume into multiple sets in order to create different volumes of word vectors that contain different vocabulary sizes. These volumes have changed after pre-processing; Table (5.1) summarize these volumes

Table (5.1): Wikipedia text volumes after pre-processing

The original volumes	After Pre-processing volumes
304 Megabytes	259 Megabytes
608 Megabytes	512 Megabytes
1200 Megabytes	919 Megabytes
1700 Megabytes	1300 Megabytes

From Table (5.1), we notice that around 400 megabytes of the text have been removed from the corpus volume after pre-processing and it presents 23% of the total volume. This is due to the nature of the structure of Wikipedia article pages which contains a lot of symbols (like, numbers, dashes, brackets, dots, backslash, etc.). This percentage is considered a little bit high. If we consider to use stemming method in pre-processing, more data would be thrown and this will affect the number of vocabularies in the generated word vectors. Also we need a big corpus to get the word co-occurrence distribution closer to real distribution and create more rich semantic vocabularies. For these reasons, we don't consider to use stemming or any other pre-processing methods that largely affects the text volumes after pre-processing.

5.1.2 The experiment environment.

We performed the experiments using two different environments. The first environment was used for text pre-processing and to create the word vectors models which requires a lot of RAM. We performed our experiments on Virtual Private Server (VPS) with the following specifications; Intel Xeon E5-2620v3, Ten cores CPUs, 50 GB RAM, 1200 GB disk space, Ubuntu 17 Operating System. The second environment is used to create the classification models and create the word clusters. For this environment we used the personal laptop with the following specification: Intel Core(TM) i5-2430M CPU at 2.40GHz, four cores CPUs, 8 GB RAM, 500 GB disk space, Ubuntu 17 Operating System.

5.1.3 The experimental parameters

w2v has main parameters that affect both training speed and quality of the generated vectors. There is no universal rules-of-thumbs to tune or control these parameters. So, after researching and taking in consideration the nature and the size of the downloaded text, we initialize and experiment the w2v model with the following parameters:

- We used the *CBOW* architecture instead of skip gram (skip gram is slower but better for infrequent words).

- We set the vector *size* to 100, reasonable values are up to 300 and it depends on the computational power and the size of text volume. So due to the size of the downloaded corpus we set the vector size to 100.

- We set the *window* size (the maximum distance between a target word and words around it) to 5. This window size is applicable since more distance will yields less semantic vectors.

- We set *min_count* (the minimum count of words to consider when training the model; words with an occurrence less than this count is ignored) to 5. This count of words is applicable since the downloaded articles context are written for different topics, and this will yield more vocabularies. More count words will yield less vocabularies and thus creating poor word vectors quality.

By applying these steps, a collection of word vectors is created. Figure (5.1) illustrates a generated word vector for the word year (سنة), the vector size is 100 and these float values represent the coordinates of the adjacent words in this N (N being the size of the word vector) dimensional space.

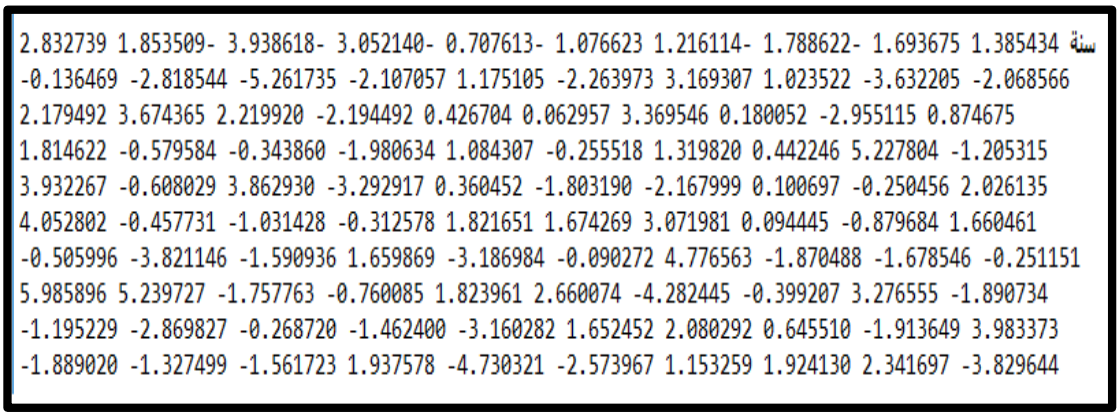

Figure (5.1): Generated word vector for word year

5.1.4 The experiment procedures.

In this section we give detail description of our experiment and provide step by step to create word vectors, so it can help others to copy our experiments and generate new results and used for their future researches.

1- We download the Arabic Wikipedia XML dump file and converted into multiple text files using wiki2text open source tool and works very well with gigabytes of XML files. Then we used cat a Linux command line to assembles different text files into one file.

2- After extracting the text file from XML file, we pre-processed the 1.7 GB text file using our written Python script and other library. See Appendix (A.1) for the source code of pre-processing task. For this task we used the VPS environment, since pre-processing large file needs a lot of computer RAM.

3- After pre-processing, we have 1.3 GB plain Arabic text. We used word2vec tool to generate word vectors from the pre-processed text file. See Appendix (A.2) for source code that creating word vectors task. For this task we used the VPS environment, since creating word vectors from gigabytes text file needs a lot of computer RAM.

4- After creating word vectors, we averaging these vectors for each word and build the classification model and created the word clusters using Python. See Appendix (A.4) for the source code of creating the classification model. For this task we used local machine environment to fitting the classification model with training and testing data.

5.2 Experimental Results and Discussion

This section describes the results of the various experiments that have been conducted.

We used the Wikimedia database dump of the Arabic Wikipedia as of on May 20, 2017 which is a collection of written articles in various fields with volume of (1.7GB). After Pre-processing (removing none Arabic character and symbols, removing the Arabic diacritics, removing the Tatweel or character elongation from the words) we produce a plain Arabic text with a volume of (1.3GB). We used VPS computer (the first environment) as presented in Section 5.2 to perform the pre-processing tasks. Table (5.2) shows the execution time in minutes for pre-processing different volumes of Wikipedia dump database.

Table (5.2): The execution time of pre-processing tasks

Text Volume	Execution Time (Minutes)
304 Megabytes	0.45098559856414794
608 Megabytes	0.7984124342600505
1200 Megabytes	1.7527989347775776
1700 Megabytes	4.128003748257955

Table (5.2) shows the execution time for pre-processing the downloaded text which take around 4 minutes. This execution time interval is considered reasonable and fast, taking into consideration the big volume of the downloaded text (1.7GB) and the different transformations that are performed on the text.

Also, we used VPS environment to create word vectors that are used to create the classification features and used in the classification model. Table (5.3) shows the

execution time for creating word vectors, as time increases along with increasing the text size. Figure (5.2) is and experiment snapshot that depicts logs information resulted from creating word vectors.

Table (5.3): The execution time of creating different word vectors

Text Volume	Execution Time (Minutes)
259 Megabytes	2.5690334320068358
512 Megabytes	5.317067114512126
919 Megabytes	9.832969482739767
1300 Megabytes	12.113544952869415

```
/usr/bin/python2.7 /root/PycharmProjects/w2v/make_vector.py
Starting training using file 1700m-p.txt
Words processed: 130900K     Vocab size: 34842K
Vocab size (unigrams + bigrams): 18860628
Words in train file: 130902867
Starting training using file 1700m-phrases.txt
Vocab size: 846693
Words in train file: 103491921
Alpha: 0.000390  Progress: 98.44%  Words/thread/sec: 165.71k
('The Totatl Execution Time in Minutes is: ', 12.113544952869415)
Process finished with exit code 0
```

Figure (5.2): Experiment logs information of creating word vectors results

Figure (5.2) shows the logs of two process: process 1; log information of creating word phrases using word2phrase package. Process 2; log information of creating the word vectors. These logs contain the total number of words in the text file before and after processing. Also this Figure shows the total time for creating these process.

We used the k-folds cross validation to spilt the dataset into k different folds (or subsets). To train our dataset we use k-1 subsets and leave the last subset as test data. To calculate the accuracy of the classification model, we take the average of the previous subsets accuracies. For example, let's divide the following dataset into 5 subsets (k=5) as it shown in Figure (5.3).

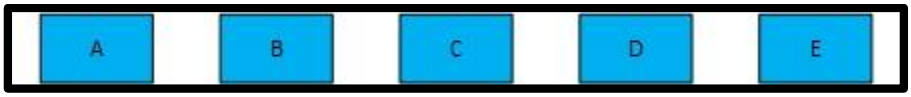

Figure (5.3): Dataset divided in 5 part

According to k-fold technique, we need to apply the classifier algorithm on k folds where we leave one of the parts for testing and use the others for training. See how this looks like for k = 5 in Figure (5.4). To calculate the overall classifier accuracy, we calculate the accuracy for every fold and takes the average of these folds.

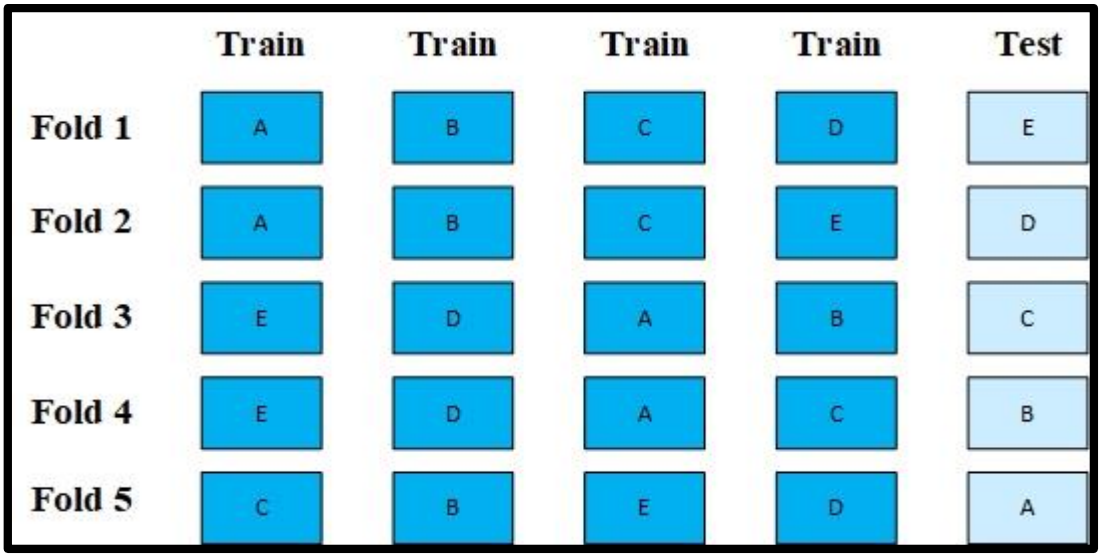

Figure (5.4): 5 k-fold divided dataset

By using the k-folds we avoid our model not being under fitting (i.e., the model does not perform poorly on the training data) and it is also not being over fitting (i.e., the model performs well on both; the training data and the testing data). To determine whether a

classification model is under fitting or overfitting the training data we look at the predication error on the training data as well as on the testing data.

We choose the class labels based on the varying content of Wikipedia articles and based on the generated word phrases like: جزيئات_الماء (water molecules), العناصر_الكيميائية (chemical elements), المسطحات_ المائية (Waterbodies), المركبات_ الكيميائية (chemical compounds) which are generated during creating word vectors, so we have chosen the following 16 labels : عنصر كيميائي (chemical element), مركب كيميائي (chemical compound), مسطحات مائية (waterbodies), مصطلح جيولوجي (geological term), مصطلح فلكي (astronomical term), دولة آسيوية (Asian country), دولة أفريقية (African country), دولة أوروبية (Europe country), دول أمريكا الجنوبية (south America country), دول أمريكا الشمالية (north America country), مصطلح طبي (medical term), مصطلح سياسي (political term), مصطلح اقتصادي (economical term), مصطلحات قانونية (legal terms), مصطلحات أدبية (arts terms), مصطلحات فزيائية (physical terms).

We have executed our experiments with varying volumes of word vectors. The k-folds values and different sets of training labelled data are used in order to create high accurate classification models that are able to classify new words into their categories. Also, we used unlabelled data to check the effectiveness of our approach and prove that the generated classification model is able to predict unseen examples.

We experiment with vector sizes of 251MB, 414MB, 643MB and 781MB. Also we experiment with different volumes of labelled training data and different k-folds values as well as we experiment once with unlabelled testing data (never seen before). All these results are summarized in Table (5.4)

Table (5.4): Summary of all experiments

Vectors Sizes / Training Data Volumes			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**

Table (5.4) 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.

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 unlabeled data) scores manually. The 15 words are: راديوم_ سينت مارتن _ سانت مارتن _النمسا _ سلطنة عمان _ الزهرة_ تايوان _ المشتري كوريوم_ _ القولون _ مناعة_ تجاذب _ الذبذبة_ بحر العرب _بحر مرمرة. For larger testing data (unlabeled 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 (5.5) shows the printed confusion matrix for 15-word testing unlabeled data.

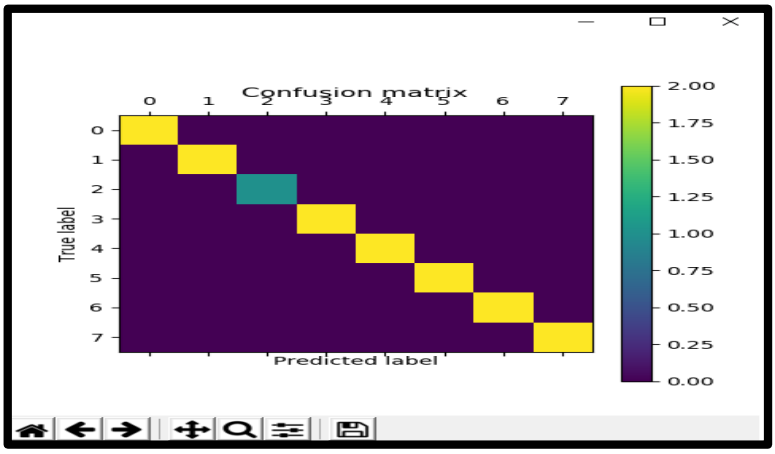

Figure (5.5): Confusion matrix for 15-word testing unlabeled data

As we see from Figure (5.5), 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.

5.4 Summary

We have presented in details the setup aspects of the performed experiments in order to create high quality word clusters. To perform our experiments, we have used the Wikimedia database dump of Arabic Wikipedia articles as of May 20, 2017. The dataset volume is about 1.7GB and has become 1.3GB after pre-processing. We have divided the dataset into multiple sets to create word vectors with different vocabulary sizes. We performed the experiments using two different environments: a cloud based and remote virtual private server environment to create word vectors and pre-processing, and a local machine environment to generate word clusters and evaluate the classification models. We have implemented the different parts of our model using Python (version 2.7). There are four main parts based on the approach as presented in Chapter 4: data pre-processing, creating word vectors, generating the classification model and creating word clusters, and word clusters evaluation.

After fitting the classification model with training data and test data, a sixteen word clusters have been created. The average evaluation scores of the classification model are promising (above 85%) and indicates that the classification models are not being over fitting or under fitting with training and test data.

Chapter 6
Conclusion and Future Work

6.1 Conclusion

We have built an approach to create word clusters from large Arabic plain text based on similarity among words which reflects the semantic relationship among them. These clusters have high intra-cluster similarity (words within a cluster are similar) and low inter-cluster similarity (words from different clusters are dissimilar).

There are four main parts of the approach as presented in Chapter 4: Part 1 is the data pre-processing which includes collecting the dataset and performing pre-processing on it. Part2 involves creating word vectors using word2vec model and other method. Part 3 involves generating the classification model and creating word clusters using Pipeline method and Extra tree classifier. Part 4 involves evaluating the generated word clusters using the most common metrics such as precision, recall, f-measure as well as confusion matrix.

We have implemented these parts of the approach using Python (version 2.7) and based on this implementation, we have performed a set of experiments. The experiments results show the effectiveness of our approach to create word clusters from a large plain Arabic text. Also, the classification results show that the extracted features from the word vectors have empowered the classification models and achieved accuracy, precision, recall and F-measure with higher than 85%. In addition to that, the classification model results indicate that the classification model is not being under fitting (i.e., the model does not perform poorly on the training data) and it is also not being over fitting (i.e., the model performs well on both; the training data and the testing data).

Finally, the proposed approach can be used efficiently and accurately to create word clusters from large Arabic plain text in any domain. It is suitable for various natural language processing tasks such as, part of speech tagging, online dictionaries, named entity recognition. In addition to that, the generated word vectors can be updated with new

vocabularies without the need of recreating these word vectors again. This helps to provide NLP applications and tasks with in-demand semantic data.

6.2 Future Works

Our work can be extended to include creating word vectors from trigram and n-grams words from large plain text. It can be applied for sentence level or document level using another neural network tool called doc2vec. Also, the generated word vectors can be used for other Arabic NLP application and tasks such as, name entity recognition, and query search expansion. These applications and uses need to be checked experimentally based on suitable models and approaches.

For fast processing, we can extend our work to use a parallel processing environment based on, e.g., Map/Reduce parallel programming model to speed up the process of creating word vectors specially with larger volumes of data (big data). We can use Hadoop (a Map/Reduce an open-source framework) to pre-process large dataset and create large word vectors.

The results of our approach are encouraging and show that high quality word vectors created with high classifier performance and thus creating high quality word clusters. Therefore, there is a need to investigate further techniques for creating word clusters from large unstructured text with less computational resources and apply them to interesting applications.

The Reference List

Abu Tair, M. M., & Baraka, R. S. (2013). Design and evaluation of a parallel classifier for large-scale Arabic text. *International Journal of Computer Applications, 75*(3).

Akata, Z., Reed, S., Walter, D., Lee, H., & Schiele, B. (2015). *Evaluation of output embeddings for fine-grained image classification.* Paper presented at the Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on.

Al-Anzi, F. S., & AbuZeina, D. (2017). Toward an enhanced Arabic text classification using cosine similarity and Latent Semantic Indexing. *Journal of King Saud University-Computer and Information Sciences, 29*(2), 189-195.

Al-Khalifa, H., & Al-Wabil, A. (2007). *The Arabic language and the semantic web: Challenges and opportunities.* Paper presented at the The 1st int. symposium on computer and Arabic language.

Al-Shalabi, R., & Obeidat, R. (2008). *Improving KNN Arabic text classification with n-grams based document indexing.* Paper presented at the Proceedings of the Sixth International Conference on Informatics and Systems, Cairo, Egypt.

Al-Zoghby, A. M., Ahmed, A. S. E., & Hamza, T. T. (2013). Arabic Semantic Web Applications–A Survey. *Journal of Emerging Technologies in Web Intelligence, 5*(1), 52-69.

Al Tarouti, F., & Kalita, J. (2016). *Enhancing automatic wordnet construction using word embeddings.* Paper presented at the Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP.

Al Zamil, M. G., & Al-Radaideh, Q. (2014). Automatic extraction of ontological relations from Arabic text. *Journal of King Saud University-Computer and Information Sciences, 26*(4), 462-472.

Alghamdi, R., & Alfalqi, K. (2015). A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA), 6*(1).

Alkoffash, M. S. (2012). Automatic Arabic Text Clustering using K-means and K-mediods. *International Journal of Computer Applications, 51*(2).

Alotaibi, S., & Anderson, C. (2017). Word Clustering as a Feature for Arabic Sentiment Classification. *IJ Education and Management Engineering*, 1-13.

Amrouche, A., Falek, L., & Teffahi, H. (2017). Design and Implementation of a Diacritic Arabic Text-To-Speech System. *International Arab Journal of Information Technology (IAJIT), 14*(4).

Asia Pacific Forum. (2005). Similarities between countries. Retrieved from https://www.eduhk.hk/apfslt/v6_issue2/foreword/foreword4.htm

Baker, L. D., & McCallum, A. K. (1998). *Distributional clustering of words for text classification.* Paper presented at the Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval.

Barbara, R. (2000). Latent Semantic Indexing: An overview. *INFOSYS, 240*.

Baroni, M., Dinu, G., & Kruszewski, G. (2014). *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.* Paper presented at the ACL (1).

Bloehdorn, S., Cimiano, P., & Hotho, A. (2006). Learning ontologies to improve text clustering and classification *From data and information analysis to knowledge engineering* (pp. 334-341): Springer.

Chennoufi, A., & Mazroui, A. (2017). Morphological, syntactic and diacritics rules for automatic diacritization of Arabic sentences. *Journal of King Saud University-Computer and Information Sciences, 29*(2), 156-163.

Denkowski, M. (2009). A survey of techniques for unsupervised word sense induction. *Language & Statistics II Literature Review*, 1-18.

Duwairi, R. M. (2006). Machine learning for Arabic text categorization. *Journal of the American Society for Information Science and Technology, 57*(8), 1005-1010.

Eldos, T. M. (2003). Arabic text data mining: A root-based hierarchical indexing model. *International Journal of Modelling and Simulation, 23*(3), 158-166.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). *A density-based algorithm for discovering clusters in large spatial databases with noise.* Paper presented at the Kdd.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning, 63*(1), 3-42.

Heap, B., Bain, M., Wobcke, W., Krzywicki, A., & Schmeidl, S. (2017). Word Vector Enrichment of Low Frequency Words in the Bag-of-Words Model for Short Text Multi-class Classification Problems. *arXiv preprint arXiv:1709.05778*.

Hersh, W., Buckley, C., Leone, T., & Hickam, D. (1994). *OHSUMED: an interactive retrieval evaluation and new large test collection for research.* Paper presented at the SIGIR'94.

Hotho, A., Staab, S., & Stumme, G. (2003). *Ontologies improve text document clustering.* Paper presented at the Data Mining, 2003. ICDM 2003. Third IEEE International Conference on.

Huang, A. (2008). *Similarity measures for text document clustering.* Paper presented at the Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand.

Istizada. (2017). complete-list-of-arabic-speaking-countries-2014. Retrieved from http://istizada.com/complete-list-of-arabic-speaking-countries-2014

Kohavi, R., & Provost, F. (1998). Glossary of terms. *Machine Learning, 30*(2-3), 271-274.

Lilleberg, J., Zhu, Y., & Zhang, Y. (2015). *Support vector machines and word2vec for text classification with semantic features.* Paper presented at the Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015 IEEE 14th International Conference on.

Ma, L., & Zhang, Y. (2015). *Using Word2Vec to process big text data.* Paper presented at the Big Data (Big Data), 2015 IEEE International Conference on.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). *Learning word vectors for sentiment analysis.* Paper presented at the Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.

Mikolov, T., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*.

Moh'd A Mesleh, A. (2007). Chi square feature extraction based svms arabic language text categorization system. *Journal of Computer Science, 3*(6), 430-435.

Nabil, M., Atiya, A. F., & Aly, M. (2015). *New Approaches for Extracting Arabic Keyphrases.* Paper presented at the Arabic Computational Linguistics (ACLing), 2015 First International Conference on.

Nadbordrozd. (2016). Text Classification With Word2Vec.

Pennington, J., Socher, R., & Manning, C. D. (2014). *Glove: Global vectors for word representation.* Paper presented at the EMNLP.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.
Rosario, B. (2000). Latent semantic indexing: An overview. *Techn. rep. INFOSYS, 240*, 1-16.
slideplayer. (2018). Example of Partitional Clustering Retrieved from http://slideplayer.com/slide/5946179/20/images/7/Partitional+Clustering.jpg
Wei et al. (2015). A semantic approach for text clustering using WordNet and lexical chains. *Expert Systems with Applications, 42*(4), 2264-2275.
Wikipedia. (2017). Wikipedia:Policies and guidelines. Retrieved from https://en.wikipedia.org/wiki/Wikipedia:Policies_and_guidelines
Wu, Y.-C. (2014). A top-down information theoretic word clustering algorithm for phrase recognition. *Information Sciences, 275*, 213-225.
Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks, 16*(3), 645-678.
Zerrouki, T. (2010). Pyarabic, An Arabic language library for Python. Retrieved from https://pypi.python.org/pypi/pyarabic/ 2010

Appendix A: The Source Code

In this appendix we present the source code of the various parts of our approach.

```
# -*- coding: utf-8 -*-
import re
import codecs
import pyarabic.araby as araby
import timeit

def read_file():
    f = codecs.open("3g.txt", "r", encoding='utf8')
    data = f.read()
    f.close()
    return data

def write_file(data):
    filew = codecs.open('3g-p.txt', 'w', encoding='utf8')
    filew.write(data)

    filew.close()

def normalize_data(data):
    regex = ur'[\u0621-\u063A\u0641-\u064A]+'
    return " ".join(re.findall(regex, data))

def strip_tatweel(text):
    reduced = araby.strip_tatweel(text)
    return reduced


def strip_tashkeel(text):
    reduced = araby.strip_tashkeel(text)
    return reduced

start_time = timeit.default_timer()
data = read_file()
remove_tashkeel = strip_tashkeel(data)
remove_tatweel = strip_tatweel(remove_tashkeel)
normalized = normalize_data(remove_tatweel)
write_file(normalized)
elapsed = timeit.default_timer() - start_time
InMinutes = elapsed / 60
print ("The Total: Execution Time in Minutes is: ", InMinutes)
```

Figure (A.1): The pre-processing Source Code

```
from gensim.models.keyedvectors import KeyedVectors
import word2vec
import timeit
start_time = timeit.default_timer()

word2vec.word2phrase('3g-p.txt', '3g-phrases.txt', verbose=True)

word2vec.word2vec('3g-phrases.txt', '3g.bin', size=100, verbose=True)

elapsed = timeit.default_timer() - start_time

InMinutes = elapsed / 60

word2vec.word2clusters('3g-p.txt', '3g-clusters.txt', 100, verbose=True)

model = KeyedVectors.load_word2vec_format('3g.bin', binary=True)

model.save_word2vec_format('3g-vectors.txt', binary=False)

print ("The Total Execution Time in Minutes is: ", InMinutes)
```

Figure (A.2): Creating word vectors

```
# -*- coding: utf-8 -*-
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.ensemble import ExtraTreesClassifier
import data
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold


# Store the word vectors into dictionary

with open("304m2-vectors.txt", "rb") as lines:

    w2v = {line.split()[0]: np.array(map(float, line.split()[1:]))

           for line in lines}


# build the features, by averaging the word vectors for all vectors in a
text

class MeanEmbeddingVectorizer(object):

    def __init__(self, word2vec):

        self.word2vec = word2vec

        self.dim = len(word2vec.itervalues().next())

    def fit(self, X, y):

        return self

    def transform(self, X):

        return np.array([

            np.mean([self.word2vec[w] for w in words if w in
self.word2vec]

                    or [np.zeros(self.dim)], axis=0)

            for words in X

        ])
```

Figure (A.3): Creating the classification features

```
# classify the vectors using Extra Trees classifier
model = Pipeline([
    ("word2vec vectorizer", MeanEmbeddingVectorizer(w2v)),
    ("extra trees", ExtraTreesClassifier(n_estimators=200))])
xx = data.get_data('train_100.txt')
yy = data.label_data('label_100.txt')
X = np.array(xx)
y = np.array([yy]).reshape(100)
kf = KFold(n_splits=3, shuffle=True, random_state=42)
kf.get_n_splits(X)
for train_index, test_index in kf.split(X):
    data_train, data_test = X[train_index], X[test_index]
    target_train, target_test = y[train_index], y[test_index]


model.fit(data_train, target_train)
prediction = model.predict(data_test)
print(pd.DataFrame({'words': data_test, 'prediction': prediction}))
expected = target_test
predicted = model.predict(data_test)
cm = confusion_matrix(target_test, prediction)
plt.matshow(cm)
plt.title('Confusion matrix')
plt.colorbar()
plt.ylabel('True label')
                          plt.xlabel('Predicted label')
plt.show()
print 'The Model Accuracy: ', accuracy_score(target_test, predicted)
target_names = ['class 1', 'class 2', 'class 3', 'class 4', 'class 5',
                'class 6', 'class 7', 'class 8', 'class 9', 'class 10']
print '\nClasification report:\n', classification_report(target_test, prediction,
target_names=target_names)
```

Figure (A.4): Creating the classification model and word clusters