جامعة بيرزيت

BIRZEIT UNIVERSITY

Faculty of Engineering and Technology

Master Program in Computing

Thesis report

"Towards Automatic Construction of Arabic Synonyms"

" نحو بناء مكنز مترادفات للغة العربية الياً "

Student Number:1105486

Supervised by:

Dr. Mustafa Jarrar

This thesis was submitted in partial fulfillment of the requirements for the Master's Degree in Computing from the Faculty of Engineering and Technology at Birzeit University, Palestine

August, 2015

جامعة بيرزيت
BIRZEIT UNIVERSITY

Building Arabic Thesaurus Automatically

By Eman Naser

Approved by the thesis committee

Dr. Mustafa Jarrar, Birzeit University

…………………………………………………………………

Dr. Bassem Sayrafi, Birzeit University

…………………………………………………………………

Dr. Nabil Arman, Palestine Polytechnic University (External)

…………………………………………………………………

Date Approved:

Abstract

The world has a data revolution, which caused a huge volume of data stored in documents in different languages. This creates an important demand for developing Cross-lingual resources to serve NLP applications, in order to understand, retrieve, translate, summarize such large amounts of texts. Thesaurus and WordNet are examples for cross-lingual resources, which become core components in modern NLP applications, specially to support multilingualism.

Although there are several thesauruses for Arabic, the majority are messy - instead of providing accurate sets of synonyms for a given word, they provide "near" synonyms and general/specific words. For example, according to the Google's Arabic thesaurus, the synonyms of the words " دولة " are {بلد, قطر, وطن, الريف, ريف}. Here the underlined words are wrong, as they are not really synonyms.

In this thesis we build an Arabic thesaurus automatically and map this thesaurus to the English WordNet. That is, the result will be a set of Arabic synsets mapped into WordNet synsets, as $\{a_1, a_2, \ldots, a_n\} := \{wn_1, wn_2, \ldots, wn_m\}$. To do this, we will first implement SynsetGenerator algorithm for generating multilingual thesauruses which requires a set of Arabic-English bilingual dictionaries as input, then we evaluate our results and link the generated results with WordNet using Cosine similarity approach.

ملخص:

ان النمو السريع لمصادر المعلومات المتصلة بالانترنت من قواعد بيانات وغيرها ادى الى حدوث ثورة في عالم البيانات. هذا أدى الى وجود كم هائل من البيانات المخزنة في وثائق ونصوص في لغات مختلفة . حيث اصبحت من اهم القضايا التي تؤرق العالم، الامر الذي ادى الى زيادة الطلب على التطبيقات البرمجة اللغوية الذكية من فهم واسترجاع وترجمة وتلخيص هذه الكميات الكبيرة من النصوص والتي تتطلب دقة في معالجتها، خاصة الجانب اللغوي. تعتبر المكانز اللغوية من الأمثلة على المصادر اللغوية المهمة ومن المكونات الأساسية للعديد من التطبيقات الذكية خاصة لدعم تعدد اللغات.

على الرغم من وجود العديد من المكانز الخاصة باللغة العربية، الا ان غالبيتها غير دقيقة خاصة وان علاقة المترادفات فيها تقريبة وهي اقرب لان تكون علاقة عامة بين الكلمات. مثلا قاموس غوغل للعربية، والمرادفات للكلمة "دولة" هي {بلد، قطر، وطن، الريف، ريف}. هنا الكلمات التي تحتها خط خاطئة، لأنها ليست مرادفات صحيحة.

تهدف هذه الأطروحة الى بناء مكنز للغة العربية بطريقة آلية، ومن ثم ربط مجموعات المترادفات في هذا المكنز بمقابلاتها في شبكة الدلالية للكلمات الانجليزية (English WordNet) بمعنى آخر، نأخذ الكلمات العربية المرادفة التي تم بناءها من المكنز و من ثم ربطها بالمترافات (English WordNet) ، مثلا {ع$_1$، ع$_2$، ...... ع$_ب$} =: {ن$_1$، ن$_2$،......ن$_ج$}. لعمل ذلك، قمنا بتطبيق خوارزمية SynsetGenerator لبناء المكنز بطريقة الية ، حيث تحتاج الخوارزمية الى مجموعة من القواميس ثنائية اللغة عربي - إنجليزي كمُدخل، وتم ربط النتائج مع (English WordNet) وتقيمها باستخدم مبدأ تشابه جيب التمام (Cosine Similarity).

Acknowledgements

I wish to express my sincere gratitude to My Supervisor Mustafa Jarrar for his great help, close supervision, and valuable input and feedback in all stages of this research.

I wish to express my thankful to the Eng. Mohammed Darawisha for his help in technical and programming parts of this thesis.

I wish also to express my thankful to Rami Asia and Faeq Alrimawi from Sina institute for their help in my Thesis Experiment as well as providing datasets, and for proofreading parts of my work.

Last but not least, I'm very thankful to my parents, my husband and my childes for the great support to give me during my research.

List of Contents

List of Figures

List of tables

List of Abbreviations:

NLP	Natural Language Processing
ANLP	Arabic Natural Language Processing
WN	WordNet
EWN	English WordNet
AWN	Arabic WordNet
VSM	Vector Space Model
HBIL	Homogeneous Bilingual Dictionary
AN	Adjective-Noun
SV	Subject-Verb
VO	Verb-Object
SW	Spanish Word
MADA	Morphological Analysis and Disambiguation for Arabic
ADO	Active X Data Object
GUI	Graphical User Interface

Chapter 1

1. Introduction

In this chapter we present a brief introduction regarding research conducted and its motivation. Then we summarize our research goals, contributions, and an overview of the thesis structure.

1.1. Scope and motivation

Arabic language is a Semitic language that most Muslims around the world speak. Moreover, it is the official language for more than 300 million in the Arabic world. It is a structural and derivational language where morphology has an important role [1]. Nowadays, research in Natural Language Processing (NLP) has reached advanced stages, which gives computers the ability to understand the way humans learn and use language. NLP has three language models: (1) rule based which uses a predefined set of rules (knowledge) such as derivational rules, inflectional rules, grammatical rules, etc. (2) statistical based which uses and calculates the probabilities of what normally people write or say, and (3) hybrid models which combines both rule based and statistical based.

Most NLP applications (e.g., translation, spelling checkers, question answering, and summarization) work by parsing words and sentences based on a language model and typically use lexical resources (e.g., thesaurus and WordNet) of the target language, which become the core of these applications[2].Arabic Natural Language Processing (ANLP) has become a popular area of research, although most of applications developed by non-Arabic speakers which focus on tools to enable non-Arabic speakers make sense of Arabic texts [2].

In the last few years, Arabic cross-lingual resources such as Arabic Thesaurus and English WordNet (EWN), which became the core of ANLP, are increasing in order to give correct and precise answers when using ANLP applications such as translating, summarizing, and retrieving information in Arabic for Arab speakers [1].Surprisingly, little has been done in the field of computerized Arabic lexical resources, which made an interesting area for many researchers.

In order to understand what is Arabic thesaurus and EWN let us start from this point; two words are said to be synonym if they can be interchanged in a context without changing the meaning [3] such as { نظم، رتب } in Arabic or {arrange, organize} in English. For example, if we can exchange the word نظم with the word رتب in a sentence like:

قام الموظف بترتيب الاوراق التي كانت مبعثرة في مكتبه

قام الموظف بتنظيم الاوراق التي كانت مبعثرة في مكتبه

Because we were able to change these two words in the same sentence (i.e., context) without changing the sentence meaning, then these two words can be considered as synonyms. **Synonyms** are defined in [11] as "*two expressions are synonymous if the substitution of one for the other never changes the truth value of a sentence in which the substitution is made*". The semantically similar words can be interchanged in the contexts more while the semantically dissimilar words cannot. The relation is said to be *symmetric*: if x is synonym to y, then y is synonym to x [11]. The synonymy is a lexical relation between the word forms which are *not transitive* [11].

Synonym Sets (Synsets) are collection of synonym words that can be used interchangeably in the same context.

Notice that one of these words might be used in other sentences/contexts to provide different meaning. Other examples of Arabic synonyms are {عام, سنة} also {منتدى, نادي}.

Notice that there should be at least one context where words can be used interchangeably to be called synonyms.

There is a case where one word can replace the other but not vice versa, as in the following sentences, but in such cases these words cannot be called synonyms, as in most cases, one is a generalization of the other.

رأيت الرجل الذي مر من هنا حاملا ابنه

رايت الانسان الذي مر من هنا حاملا ابنه

In this case, the word انسان is not as (arrange, organize) := (رتب, نظم) to the word رجل although we were able to replace them; because the second sentence is more general than the first , as انسان is more general than رجل .

Thesaurus is defined as *a bag of words which is organized to help in finding words related to a core concept but having different shades of meaning (connotations)*[4]. Moreover, Oxford English dictionary defines a thesaurus as *a book that lists words in groups of synonyms and related concepts*\[5]. In Arabic, a thesaurus is normally called " مكنز " [6]. It should be noted that unlike our definition of synonyms described above, the notion of synonyms used in typical thesauruses is not well defined and thus it merely refers to related words.

WordNet (WN) is defined as a network of lexicalized concepts (synsets a set of synonyms) which are sets of word meanings considered being synonymous within a context. synsets are connected by several semantic relations (hyponymy, meronymy) [1], the hyponymy relation is a semantic relation between word meanings.

1.2. Problem statement and thesis goals

This thesis aims to investigate building Arabic thesaurus automatically - derived from existing Arabic-English bilingual dictionaries, then map the derived sets of synonyms to WN synsets, as the following:

$$\{a_1, a_2, \ldots, a_n\} := \{e_1, e_2, \ldots, e_k\} := \{wn_1, wn_2, \ldots, wn_m\}$$

Where a_n refers to an Arabic word, e_k refers to an English word, and wn_m is an WN's word. The sign := means that there is semantic equivalency between the two synsets. To build the Arabic thesaurus our approach has to accomplish the following goals:

1. Implement SynsetGenerator algorithm [7]for the purpose of generating the Arabic and English synonyms. Then, evaluate whether the implementation presents satisfying results (i.e., whether it generates synonyms as expected) using the existing Arabic and EWNs resources.
2. Map the generated Arabic and English synonyms into WN synsets.

3. Evaluate the reliability of our approach by finding how much of the generated synonyms can be mapped into the EWN.

1.3. Summary of contributions

The thesis contributions can be summarized as the following:

- Implemented the SynsetGenerator algorithm, and used it to derive (given the Arabic bilingual dictionaries) a large set of Arabic and English synonyms.
- Evaluated whether our implementation is working correctly (i.e., whether it generates synonyms as expected) using the existing Arabic and EWNs resources.
- Mapped the generated Arabic words' synonyms to their equivalent WN synsets. This is done by listing all WN synsets which have common English words synsets and giving each match a weight. The highest WN synset weight is then mapped to the Arabic word synsets.
- Evaluated the reliability of our approach which is done through finding how much of the generated synonyms can be mapped into the EWN.

1.4. Overview of thesis structure

In chapter two we will present related work to our research. First, we will give an introduction about NLP, and then we will present some important definitions and related resources such as thesaurus, WN, ontology, and linguistic ontology. Most of the chapter will be dedicated to related work done in Automatic Thesaurus/WN Construction methods. We will discuss two methods in building automatic thesaurus. First method is "Building A Cross-lingual Relatedness Thesaurus Using A Graph Similarity Measure". Second method is "Automatic Thesaurus Construction". We will also discuss five approaches for building WN automatically. First approach introduces the AWN Project. Second approach called "Semi-Automatic Development of FarsNet; The Persian WN". Third approach called "Combining Multiple Methods for The Automatic Construction of Multilingual WNs automatically". Fourth approach is Building "Polish WordNet PolNet project". Last approach is about "Building Czech WordNet".

After that in chapter three we will describe the algorithm idea and methodology that we followed to achieve thesis goals. Firstly, we introduce the idea and methodology of the algorithm to create Arabic thesaurus file (Arabic-English synsets).Secondly, we introduce the implementation of the algorithm.

Then in chapter four we introduce the testing and mapping algorithms' methodologies. Firstly, we describe the testing algorithm methodology and implementation in order to find the bugs in our algorithm implementation and evaluate the algorithm performance. Secondly, we introduce the mapping algorithm methodology and implementation to map the generated Arabic words synonyms to their WN equivalent synsets, the highest WN synset weight is mapped to the Arabic word synsets.

Chapter 2

2. Literature Review

2.1. Introduction

In this chapter, we discuss and distinguish between Thesaurus and WordNet (WN), and present related works done on Automatic Thesaurus/WN Construction. In section 2, we define and differentiate between thesaurus, WN, ontology, and linguistic ontologies, and explain their true value. We review the differences between Thesaurus, WN, and Ontology in section 3. In section 4, we review work done on building Thesaurus/WN automatically. In section 5, we will discuss the evaluation methods that could be used to evaluate our proposed algorithm. Finally, in section 6 we compare between the evaluation methods.

Literature discussed in this chapter has two basic ideas. The first idea is that some approaches can be valid for a certain language and not for the others. Arabic language was a challenge because of its complexities (morphological and semantic issues). The second idea is that the main goal of Arabic Thesaurus/WN automatic construction is to find the common base concept between many languages then translate it. The missing concepts are added manually. This approach did not give an accurate result and may have irrelevant synset.

Our approach takes a different direction than the earlier mentioned ideas. The strength of our approach comes from mapping the Arabic words synsets (the file which we get from the bilingual dictionary) to the EWN synsets by listing all WN synsets which have common English words with the English synset (the file which we get from the bilingual dictionary) and give each one a weight. The weight increases when the number of common English words increase. The highest WN synset weight is then mapped to the Arabic words. The following sections review other's approaches in and compare them with our approach.

2.2. The difference between thesaurus, WordNet (WN), ontology and linguistic ontologies

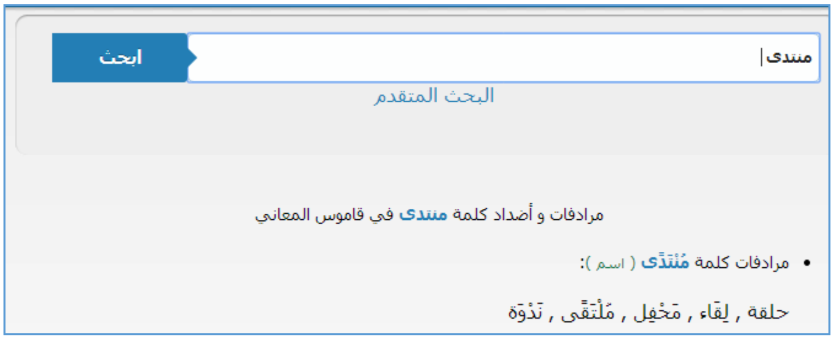
2.2.1. Thesaurus definition

Thesaurus is defined as a *bag of words that groups words related to a core concept together, but may have different meaning*[4] *as it does not specify the relationship between the words*. Oxford English Dictionary defines thesaurus as, "*A book which lists the words in groups of synonyms and related concepts*" [5]. In Arabic, a thesaurus is commonly called "مكنز" [6].

The importance of Thesaurus are as follows [4]:

- Finding words that you need to express your idea effectively, descriptively and in more interesting way.
- Avoiding the repetition of the same words.
- Recalling the word that is on the tip of your tongue.
- Broadening vocabulary through trying out new words and phrases.
- Communicating with greater confidence.
- Finding the word that suits the genre (e.g. a letter), purpose, intended audience and context of what you are writing.
- Information retrieval.

Figure 2.1 shows an online Arabic thesaurus called "المعاني". We search "منتدى" word and obtained its synonyms as shown below [8]:



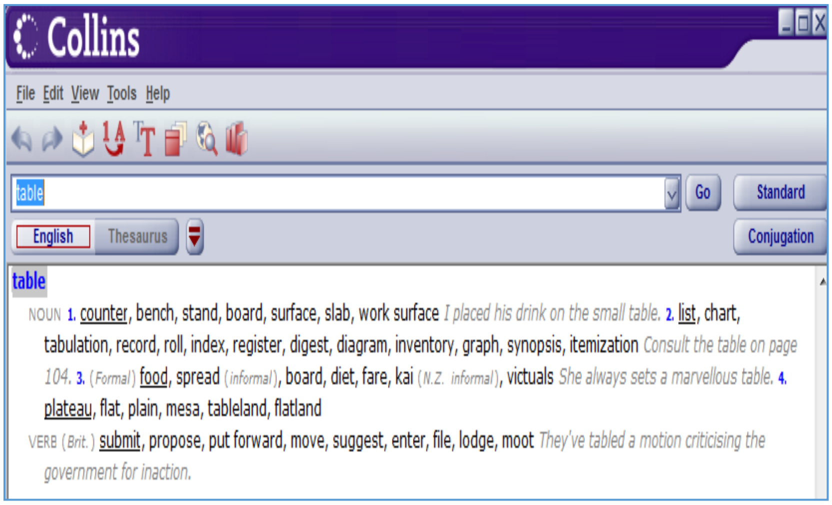
مرادفات و أضداد كلمة منتدى في قاموس المعاني

- مرادفات كلمة مُنْتَدًى (اسم):

حلقة , لِقَاء , مَحْفِل , مُلْتَقًى , نَدْوَة

Figure 2.1:المعاني online Arabic Thesaurus [8]

Figure 2.2 shows English thesaurus called “Collins”. Collins is a desktop application that has easy interfaces. We searched synonyms for the word “table” [9].



table

NOUN 1. counter, bench, stand, board, surface, slab, work surface *I placed his drink on the small table.* 2. list, chart, tabulation, record, roll, index, register, digest, diagram, inventory, graph, synopsis, itemization *Consult the table on page 104.* 3. (*Formal*) food, spread (*informal*), board, diet, fare, kai (*N.Z. informal*), victuals *She always sets a marvellous table.* 4. plateau, flat, plain, mesa, tableland, flatland

VERB (*Brit.*) submit, propose, put forward, move, suggest, enter, file, lodge, moot *They've tabled a motion criticising the government for inaction.*

Figure 2.2: Collins desktop English thesaurus [9]

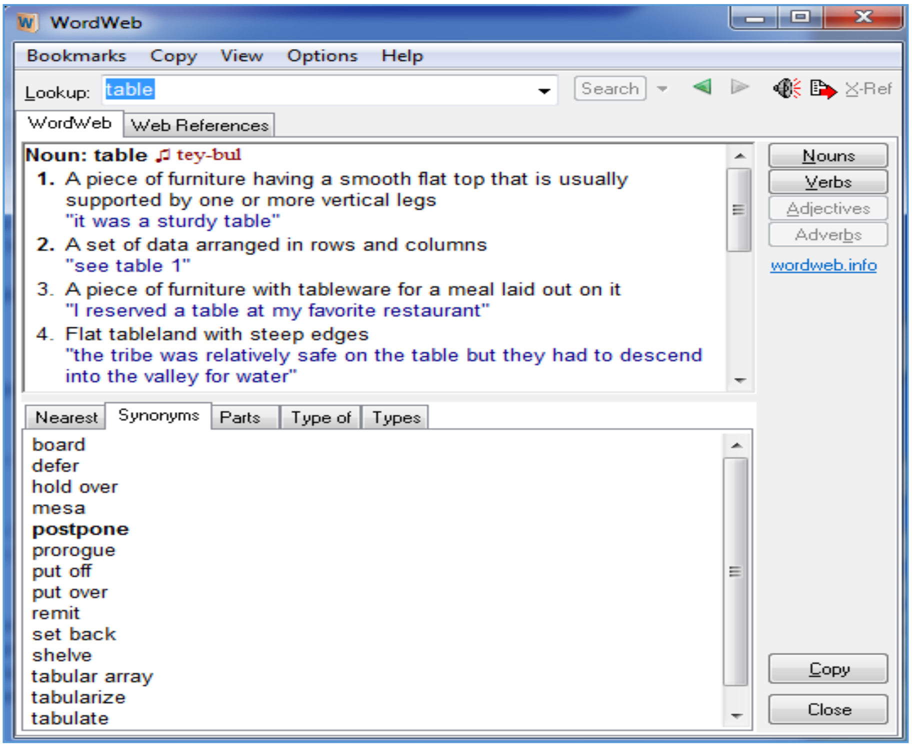
2.2.2. WordNet definition

WordNet is a lexical knowledge base for English [10], which is electronically available for free. The idea of the WN project was born by the psycholinguistic theory of human lexical memory [10]. WN organized the nouns, verbs, adjectives and adverbs into sets of synonyms, each set represents a lexical concept. These concepts (synsets) are connected by several semantic relations, such as (hyponymy and meronymy) [11].

When designing a WordNet for a language, some designers try to be as close as possible to the EWN, but they should respect the specific properties of that language [12]. While others try to build WordNet from scratch which means that the designer must apply a suitable classification criteria to the linguistic material in order to generate formal representations of concepts that consist of synsets and relations [12].

Creating and building WordNet is of importance since it provides a way for word sense disambiguation, information retrieval, automatic text classification, summarization, machine translation, and others [11].

In this section we will briefly talk about English WordNet (EWN) and Arabic WordNet (AWN). Figure 2.3 shows a desktop application called WordWeb which is a nice interface for the EWN.



Noun: table ♫ tey-bul

1. A piece of furniture having a smooth flat top that is usually supported by one or more vertical legs
   "it was a sturdy table"
2. A set of data arranged in rows and columns
   "see table 1"
3. A piece of furniture with tableware for a meal laid out on it
   "I reserved a table at my favorite restaurant"
4. Flat tableland with steep edges
   "the tribe was relatively safe on the table but they had to descend into the valley for water"

board
defer
hold over
mesa
postpone
prorogue
put off
put over
remit
set back
shelve
tabular array
tabularize
tabulate

s

Figure 2.3: WordWeb desktop English WordNet [13]

2.2.2.1. EWN definition and structure

EWN is a large lexical database, which is grouped into synonyms set (synsets), each expressing a distinct concept. The synsets are linked by semantic relations between concepts [11]. EWN is structured as a set of synsets where each synset has a unique ID called SynsetID, and a short gloss that describes the concept. For example:

SynsetID: 08283156

Synset: {Table, Tabular Array}

Gloss for the concept: A set of data arranged in rows and columns.

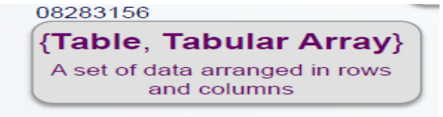


Figure 2.4: WordNet structure [14]

EWN has two properties: polysemy and synonymy, that is, words that appear within a single synset are synonymy. If the same word appears in different synsets, then they are polysemy [15]. For example:

{Table} := a piece of furniture having a smooth.

{Table, Tabular Array} := a set of data arranged in rows and columns.

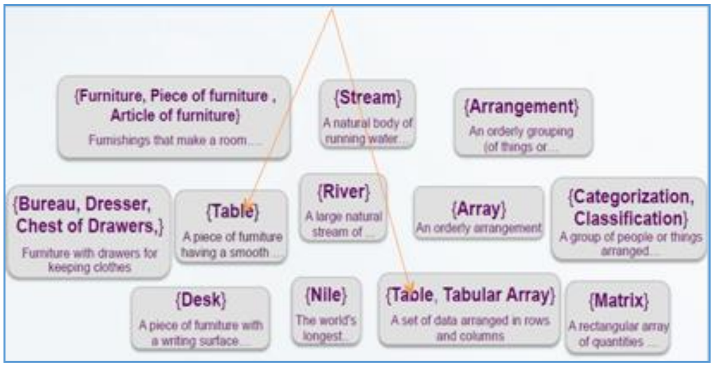

{Furniture, Piece of furniture , Article of furniture}
Furnishings that make a room....
{Stream}
A natural body of running water...
{Arrangement}
An orderly grouping (of things or ...
{Bureau, Dresser, Chest of Drawers,}
Furniture with drawers for keeping clothes
{Table}
A piece of furniture having a smooth ...
{River}
A large natural stream of ...
{Array}
An orderly arrangement
{Categorization, Classification}
A group of people or things arranged...
{Desk}
A piece of furniture with a writing surface...
{Nile}
The world's longest...
{Table, Tabular Array}
A set of data arranged in rows and columns
{Matrix}
A rectangular array of quantities ...

Figure 2.5: WordNet property [14]

Synsets are connected within semantic relations, which focus on the mapping between concepts - forming semantic network. These relations are [15, 16]:

- **Hyponymy** or "**kind of**" relation: a hierarchical semantic structure of about 16 levels. Each super level (generic concept) inherits all its features to the lower levels.
- **Meronymy**: or "**part of**" relation, each super level (generic concept) inherits all its features to the lower levels, e.g., Finger is part of hand, hand is part of arm, and arm is part of body.

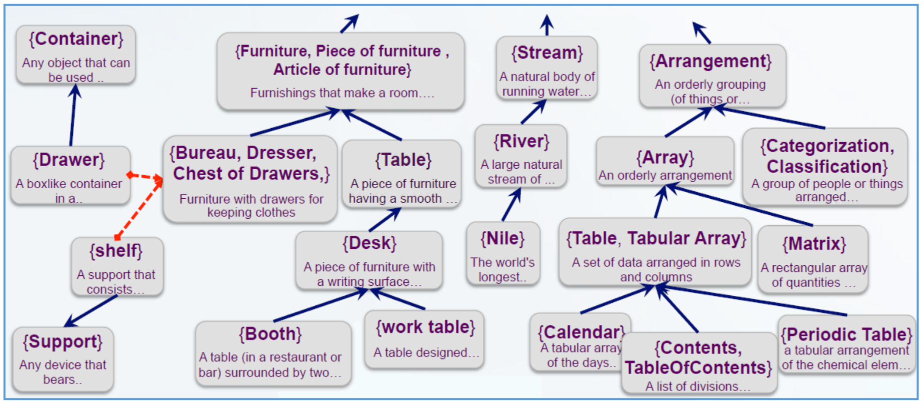

{Container}
Any object that can be used ..
{Furniture, Piece of furniture , Article of furniture}
Furnishings that make a room....
{Stream}
A natural body of running water...
{Arrangement}
An orderly grouping (of things or...
{Drawer}
A boxlike container in a..
{Bureau, Dresser, Chest of Drawers,}
Furniture with drawers for keeping clothes
{Table}
A piece of furniture having a smooth ...
{River}
A large natural stream of ...
{Array}
An orderly arrangement
{Categorization, Classification}
A group of people or things arranged...
{shelf}
A support that consists...
{Desk}
A piece of furniture with a writing surface...
{Nile}
The world's longest..
{Table, Tabular Array}
A set of data arranged in rows and columns
{Matrix}
A rectangular array of quantities ...
{Support}
Any device that bears..
{Booth}
A table (in a restaurant or bar) surrounded by two...
{work table}
A table designed...
{Calendar}
A tabular array of the days...
{Contents, TableOfContents}
A list of divisions...
{Periodic Table}
a tabular arrangement of the chemical elem...

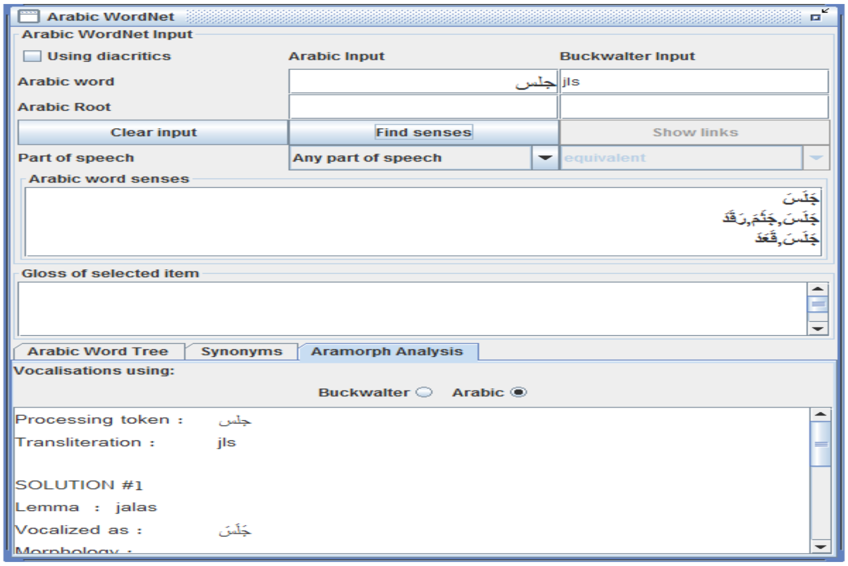
Figure 2.6 : WordNet relations [14]

2.2.2.2. AWN definition and structure

AWN is a lexical (linguistic) resource that shows and discovers the richness of Arabic language. The design and the content of AWN are based on EWN [17].

The basic idea of AWN is to use the Arabic base concepts that are defined and extended through the hyponymy relations to derive the core of the WN. The set of common concepts comes from 12 languages in EWN collected as synsets; other language-specific concepts that are not in the common base concepts are added and translated manually to the closest synsets in Arabic [17].In the literature review, we will go through some details on AWN structure.

Figure 2.7 shows an AWN desktop application [18], we searched the synsets for جلس, and we obtained two synsets {جلس،جثم،رقد}،{جلس،قعد} each synset conveys a concept. Once we click on the synset, the gloss of the concept appears. The second synset is selected as shown in Figure 2.8.



Arabic word: جلس | jls

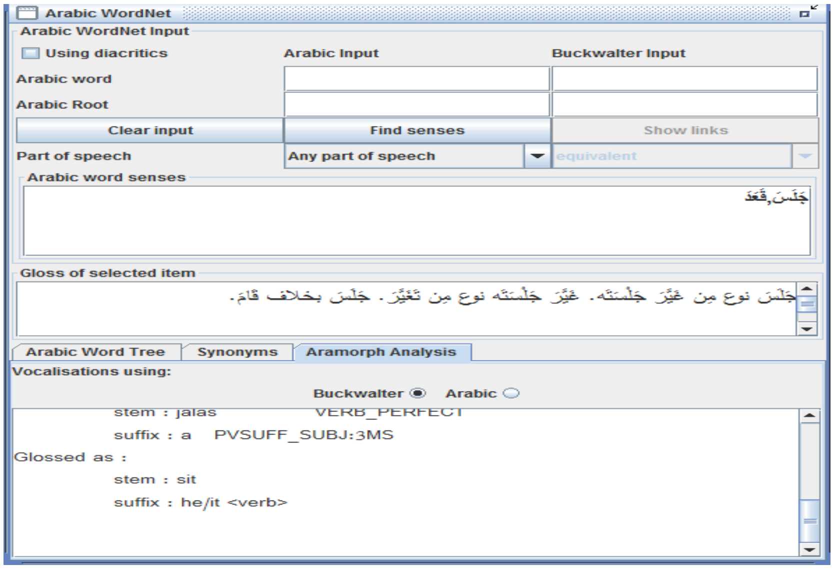
Part of speech: Any part of speech

Arabic word senses:
جَلَسَ
جَلَسَ,جَثَمَ,رَقَدَ
جَلَسَ,قَعَدَ

Vocalisations using: Buckwalter ○ Arabic ●

```
Processing token :     جلس
Transliteration :      jls

SOLUTION #1
Lemma  :  jalas
Vocalized as :         جَلَسَ
Morphology :
```

Figure 2.7: Arabic WordNet [18]



جَلَسَ,قَعَدَ

جَلَسَ نوع مِن غَيَّرَ جَلْسَتَه. غَيَّرَ جَلْسَتَه نوع مِن تَغَيَّرَ. جَلَسَ بخلاف قَامَ.

```
          stem : jalas          VERB_PERFECT
          suffix : a    PVSUFF_SUBJ:3MS
Glossed as :
          stem : sit
          suffix : he/it <verb>
```

Figure 2.8: Arabic WordNet with gloss for a selected synset [18]

AWN will be covered in more details in section 2.5 as we will discuss most resent AWN papers.

2.2.3. Ontology definition

Ontology is a shared understanding (i.e. semantics) of a certain domain, axiomatized and represented formally in a computer resource. By sharing an ontology, autonomous and distributed applications can meaningfully communicate to exchange data and make transactions interoperate independently of their internal technologies [19]. Gruber (1995) defines ontology as *"An explicit specification (which written in logic) of any model or situation) semantic structure, which encodes the implicit rules constraining the structure of a piece of reality"*[19-22].

Ontology as a term consists of two words, which are: Onto that means (exists) + logoy which means (knowledge of). In Arabic words, ontology can be defined as (الأنطولوجيا:علم الوجود بما هو موجود) [23, 24].

Ontology is of importance since it can have a shared understanding for both people and machines as it provides both specifications of semantics (i.e., meaning) of the terms and the structure, whereas XML, e.g., provides only the syntax and the database schema provides only the structure. Moreover, it provides precise and formal meanings for common vocabulary terms, but standard vocabularies as an example, do not provide that [19].

2.2.4. Linguistic ontology definition

Linguistic ontology is one of ontology types that represents the semantics of all words of a human language independently of a particular application. EWN and Arabic ontology are important examples of linguistic ontology. We will go through it in some details in the next section[7].

Linguistic ontology has the following properties [7]: (i) each word may have several concepts (Polysemy), (ii) represents common sense knowledge for a specified domain (lexical semantics), and (iii) is used for general purposes.

Linguistic Ontology is considered important since it can have a positive impact on Information search and retrieval, also word sense disambiguation such as Semantic web and web 3.0, as the query result will be improved and become meaningfully and not only a string matching. In addition, it affects machine translation performance, as it will find the exact mapping of concepts across languages. Moreover, it will enhance on data integration and interoperability. Linguistic ontology could be used as a semantic reference for many information systems [7, 14, 24].

2.2.5. Arabic ontology

Arabic Ontology can be simply defined as a tree of concepts (i.e., meanings) of all Arabic terms in Arabic language that are represented formally. That is, for each Arabic term, a set of it concepts (Polysemy) are listed. Then, these concepts are using classified semantic relationships such as subtype-of and part-of relations [10, 14, 24].

Arabic ontology is one of a long-term project started in 2010 at Sina Institute at Birzeit University. As time goes on, improvements increase continuously in both quality and quantity. Arabic Ontology defined as [7]:

"الانطولوجيا العربية هي مجموعة معاني كلمات اللغة العربية ،والعلاقات بين هذه المعاني وليس بين الكلمات".

2.2.5.1. Ontology structure

The Arabic ontology follows the same structure of WordNet, thus it can be used as an Arabic WordNet. Each Arabic word (lexical unit) has a list of glosses (concepts), each gloss describes one concept and has an ID called glossID. These concepts are connected using semantic relationships. Figure 2.6 shows the Ontology structure [19, 21]:

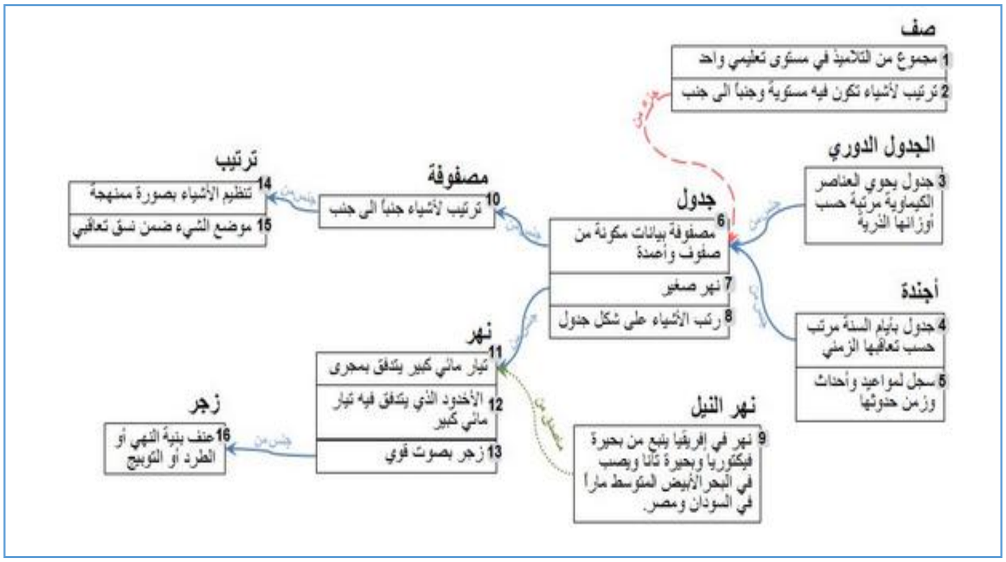

صف
1 مجموع من التلاميذ في مستوى تعليمي واحد
2 ترتيب لأشياء تكون فيه مستوية وجنباً الى جنب
الجدول الدوري
3 جدول يحوي العناصر الكيماوية مرتبة حسب أوزانها الذرية
ترتيب
14 تنظيم الأشياء بصورة ممنهجة
15 موضع الشيء ضمن نسق تعاقبي
مصفوفة
10 ترتيب لأشياء جنباً الى جنب
جدول
6 مصفوفة بيانات مكونة من صفوف وأعمدة
7 نهر صغير
8 رتب الأشياء على شكل جدول
أجندة
4 جدول بأيام السنة مرتب حسب تعاقبها الزمني
5 سجل لمواعيد وأحداث وزمن حدوثها
نهر
11 تيار مائي كبير يتدفق بمجرى
12 الأخدود الذي يتدفق فيه تيار مائي كبير
13 زجر بصوت قوي
زجر
16 عنف بنية النهي أو الطرد أو التوبيخ
نهر النيل
9 نهر في إفريقيا ينبع من بحيرة فيكتوريا وبحيرة تانا ويصب في البحر الأبيض المتوسط ماراً في السودان ومصر.
جنس من
جزء من
ما صدق من

Figure 2.9: Ontology structure [23]

Unlike WordNet, semantic relations in the Arabic Ontology include subtype relation (جنس من) (subset: A ⊆ B) in which subclass inherits all properties from the super class, as every instance in A must also be an instance of B. This relation leads to a path in the tree. In addition, instance semantic relation (ما صدق من) means it is an instance of the super class; such as Nile river class is an instance of rive class. Another semantic relation is part of relations (جزء من) which means it is a part of the super class such as صف class with GlossID := 2 is part of the class جدول with GlossID:=6 as in figure 2.6 above.

As the gloss description follows ontological rules, the most important gloss guidelines are to start with super-type class when defining the concept. To verify if it is correct or not, just put the gloss definition in the place of the class name. The example below explains the process which is followed in figure 2.9[23].

جدول : مصفوفة بيانات مكونه من صفوف و اعمدة
جدول: ترتيب بيانات جنبا الي جنب على شكل صفوف واعمدة
جدول: تنظيم بيانات بصورة ممنهجة جنبا الي جنب على شكل صفوف واعمدة.

2.3. Comparison between thesaurus and WordNet, ontology and Arabic ontology

In this section we will make a brief compression between Thesaurus, WordNet, Ontology and Arabic Ontology.

2.3.1. Can WordNet act as thesaurus?

A WN is typically more accurate than a thesaurus; because it allows us to navigate within the words due to its semantic relations. That is, related words are connected in specific concepts which allow us to measure and quantify the semantic similarity among words and concepts [16, 25].

The hypernymy relations, for example, are useful for integration into information extraction and browsing/search systems which makes it easier to find synonyms. WN becomes the base for creating multilingual WNs; this means WN becomes the base for representing the lexical knowledge between different languages [25].

However, Thesaurus is just a "bag of words" without relations between them, it just lists the related words that are synonymous within the context, and sometime there are words linked in WN that appear in the synset, but they do not appear in the same thesaurus entry. WN can be used as Thesaurus, but it is more comprehensive. By contrast, Thesaurus cannot act as WN [16].

2.3.2. Is WordNet an ontology?

Ontologies define the meaning formally, thus they are typically more accurate if compared with a WordNet. The differences between the two explained as [7, 14]: (i) meaning: WN depends on native speakers to build the relations, but Ontology depends on Scientific and philosophical findings, so it's more accurate, (ii) classification: an Ontology uses distinguishing properties when classifying a concept such as student IsA role, all types/classes are rigid, etc. and (ii) formal Specification: WN does not use formal form but Ontology is strictly formal. Arabic Ontology can be used as AWN, but its content is more accurate.

2.4. Automatic thesaurus / WordNet construction approaches

In this section, we will present two Thesaurus Construction approaches and three WN Construction approaches.

2.4.1. Building a cross-lingual relatedness thesaurus using a graph similarity measure approach

This approach [26] aims to build a German thesaurus. The main goal is to suggest words that are semantically related in a second language. For example, for a given word in one language, the German word L¨owe (lion), the method suggests ten related words: cheetah, panther, rhino (ceros), tiger, jaguar, leopard, hyena, and cub as well as the actual translation, all of which are wild animals.

This approach requires two monolingual corpora and one bilingual dictionary. Two graphs are built from the two monolingual corpora. The nodes representing words, and edges representing linguistic relations between these words. The bilingual dictionary provides seed translations which connects the nodes in both graphs.

An inter-graph node-similarity algorithm is used to discover related words. This algorithm is based on SimRank. SimRank is a recursive algorithm that is based on the idea that two nodes in a graph are similar when they are neighbors, when they have related neighboring words, or when belong to a set of correspondences between the two graphs.

Correspondences are translations (”seed translations”) provided by a dictionary. These pairs have the similarity value that is equal to one (maximum similarity). The figure below summarizes the idea.

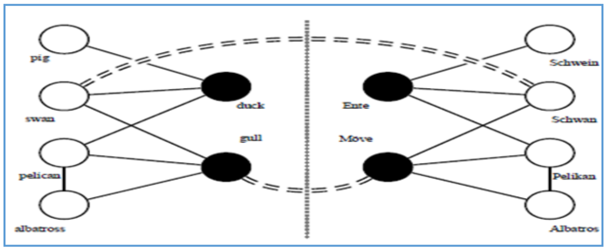


Figure 2.10 : Similarity through seed translations [26]

The double lines indicate seed translations. Let us start with the nodes duck and Ente occur in coordination's with the same nouns in the two languages; one of these swan – Schwan is a seed translation. This coordination relationship brings the similarity of duck –Ente.

This method is evaluated by three human judges, which found that this method discovers 49% of the English and 57% of the German words that are semantically related to the target words.

2.4.2. Automatic thesaurus construction approach

This approach [27] suggested a method to automatically build a thesauri using syntactically constrained distributional similarity.

The main goal is to find the distributional similarity, which is calculated through high vector space model (VSM). The dimensionality of a word with respect to the base element of the VSM is either syntactically conditioned or unconditioned i.e., if it has grammatical relation or not.

The calculations are based on the hypothesis that similar words share similar grammatical relationships and semantic contents that are done in three steps:

- Complete parsing of the sentences in corpora.
- Extract the syntactic dependencies into distinctive subsets according to head-modifier
- Determine the distributional similarity using similarity measures such as the *Jaccard* coefficient.

This approach relies on putting the terms into categories according to the grammatical relations, and on making an overlap between the top n similar words.

Similarity of words depends on how they can be interchangeable in different contexts provided that the alternation of meaning in discourse is acceptable With interchangeability of synonyms or near-synonyms in contexts. The heuristic of deriving automatic thesauri can be expressed as:

- Nouns: $U_{i,j}$ ($S_i \cap S_j$)

- Verbs: $U_{i,j}$ ($S_i \cap S_j$)

Where i and j are any two types of the dependency sets: Adjective-Noun (AN), Subject-Verb (SV) and Verb-Object (VO).

The first step of automating the Thesaurus is to build the syntactically constrained VSM. This is done using an English syntactic parser based on Link Grammar. The word space then will contain four dependency sets: Adjective-Noun (AN), Subject-Verb (SV), Verb-Object (VO) and Prepositional Phrase to Verb (RV). This step produces the latent semantic representation of words through which distributional similarity can be measured.

The Syntactic dependencies can follow a word meaning in context. The semantic requirements are mainly: determination or selector which emphasizes the semantic traits main role in the construction, dependency or dependent which add some additional trait in order to formulate the integrity of the construction [28].

A Link Grammar based parser is used to capture the main dependency categories mentioned before **(RV, AN, SV and VO)**. Each word is equipped with one or two connectors: left-pointing and/or right pointing connectors. The dependency relation between two words can be reflected by their link. The method evaluation shows that this approach can be used to build automatic thesauri with higher precision than the other traditional methods.

2.4.3. Arabic WordNet construction approach

The main goal of this approach [29] is to build an AWN based on WN. AWN will then be mapped onto WN 2.0 and EuroWN (EWN), and enable the translation to English and other languages. Arabic Base Concepts are the core of the WN which are extracted from the hyponymy relations.

The first step of AWN construction is to find the Common Base Concepts from 12 languages in WN and BalkaNet, which are translated as synsets. The missing concept are added manually. The AWN database structure has four principal tables: item, word, form and link. Item table holds information about synsets, for both English and Arabic synsets.

Word table holds information about words within synsets, for both English and Arabic words. Form table holds information about different forms of Arabic words. Link table holds the links between different synsets or words within a particular language. The following criteria is used to select the synsets in the AWN:

- Connectivity: that is, AWN should have hyperonymy/hyponymy relations as AWN synsets should match to English WN synsets .
- Relevance: that is, the most frequent concepts and lexical items are selected (from both Arabic and English).
- Generality: that is, the synsets of the generic levels of WN have high priority.

The processing within this criteria is done first from English to Arabic where for a given English synset, select all Arabic corresponding, and second, from Arabic to English where for a given Arabic word, find its senses then select English synsets corresponding for each one of these senses.

The steps of AWN contraction are:

1. The first step is to find Base Concept (BC) set from EWN and BalkaNet's CBCs manually. They focus on the most relevant terms for obtaining about 1,000 nominal and 500 verbal synsets.
2. The second step is to add two preprocessing tasks: preparation and extension.
3. In Preparation task, the bilingual resources are processed to create a homogeneous bilingual dictionary (HBIL), and then apply the morphological rules.
4. All methods used to EWN will be applied to HBIL in order to map the Arabic words/English synsets.
5. All Arabic words in bilingual resources must be normalized and lemmatized but the vowels and diacritics must be maintained.
6. Now Arabic/English synset pairs ready to be an input to the manual validation step.
7. Finally, AWN will be completed by finding all the gaps in its structure.

2.4.4. Semi automatic development of FarsNet; The Persian English WordNet approach

The main goal of this approach [30] is to build a WordNet for the Persian language called FarsNet, using a semi-automatic technique. FarsNet will include the following: (i) concepts, which have language independent base concepts extracted from BalkaNet. This makes it compatible with other WN. In addition, they have Persian base concepts extracted from the most frequent words of Peykreh and PLDB corpora, and (ii) relations, FarsNet does not cover inter-POS relations, it has inner language relationships such as synonymy, hypernymy and hyponymy, different types of meronymy, antonymy and cause which are between different synset and FarsNet synsets. Also, inter language relationships, equal-to and near-equal-to relations.

This approach requires two bilingual dictionaries, Persian-English and English-Persian and one monolingual Persian-Persian dictionary, Persian thesaurus and EWN as resources. The general methodology for this approach can be summarized as following:

- Construction of a Core-WordNet for a set of common base concepts through translating BalkaNet concepts sets BCS1 and BCS2. The most frequent and language specific concepts will be added in the next phases using electronic Persian corpora.
- Adding relational links and mix their direct semantic contexts to the common base concepts.
- Top-down extension of this core-WN by new concepts and relations.

The core of the Persian WN is constructed by adding first level hyponyms to the Base Concepts, which are more than 15,000 Persian words, and organized in 10,000 synsets of nouns, adjectives and verbs. Then this core can be semi-automatically expanded using some dictionaries, lexicons, corpora, etc.

There are two categories of conceptual relations: Taxonomic and Non-taxonomic. Both are extracted from either raw or tagged texts using the following approaches.

1. **Pattern based approach**

This approach is used to extract taxonomic relations which is in high precision but low recall in a corpus. Steps of this approach are summarized as the following:

- Find the top 1000 frequent Persian Nouns.
- Find the Wikipedia articles related to these nouns.
- Apply the Post-Processing such as eliminating the stop word, eliminating prepositional phrases for taxonomic relations, etc.

2. **Structure based approach**

This approach is used to extract both taxonomic and non-taxonomic relations. It is used for Wikipedia for the most 1,000 frequent nouns. The types of the extracted relations are not known in this method but they can be mostly found by using some extra searches on the web.

3. **Statistical based approach**

This approach is used to extract co-occurrence relations. Look for the co-occurrence words of the 500 most frequent Persian nouns in 100,000 word subset of Bijankhan corpus. By experiment, the co-occurrence threshold found to be 19.

The method evaluation shows that this approach has 72% precision in mapping Persian words to English synsets and 69% precision in mapping Persian synsets to English synsets.

2.4.5. Combining multiple methods for the automatic construction of multilingual WordNets approach

The main goal of this approach [31] is to attach Spanish word meanings to the existing WN1.5 concepts. This method requires two bilingual dictionaries Spanish/English and English/Spanish to generate a homogeneous bilingual (HBil) by merging them, a large Spanish monolingual dictionary and EWN WN1.5. In order to link the Spanish words to WN synsets, three methods are applied.

1. Class methods

These methods are used as knowledge sources coming from bilinguals and WN synsets. Hbil has 2 groups monosemous or polysemous relative to WN1.5, each has 4 different cases:

- Monosemic Criteria: These criteria apply only to monosemous English Word (EW) with respect to WN1.5.
  - Monosemic 1: A Spanish Word (SW) has only one English translation.
  - Monosemic 2: A SW has more than one English translation.
  - Monosemic 3: Several SWs have the same English translation.
  - Monosemic 4: Several SWs have different English translations.
- Polysemic Criteria: This criteria follow the four groups described in Monosemic Criteria but for polysemous English words according to WN1.5
- Variant criterion: if two or more of the EW have only one Spanish translation, then a link is between the SW and WN synset is produced.

2. Structural methods

These methods take profit of the structure of WN. it generates all combinations of English words translation from HBil in order to find the common information between the corresponding EWs in WN1.5.

- Intersection criterion: if there is at least one common synset between the WN and all EW, then a connection between SW to all common synsets.
- Parent criterion: If there is an EW synset that is a parent to the e rest of English words then a link between SW to all parent synsets.
- Brother criterion: If all EWs have synsets which are brothers respecting to a common parent then a link between SW to all cohyponym synsets.

The results of the above criteria will follow the structure: Spanish Word {list of EW} {list of synsets}

3. Conceptual distance methods

Conceptual distance provides a basis for determining closeness in meaning among word. Conceptual distance between two words depends on the length of the shortest path that connects the concepts.

4. Combining methods

The main goal for this approach is the combination of methods and sources in a way that the accuracy of the data obtained from the combined methods overcomes the accuracy obtained from the individual ones. By collecting those synsets produced by all methods described above it gives an accuracy greater than 85%.

The approach seems to be extremely promising attaching up to 75% of reachable Spanish nouns and 55% of reachable WN synsets. As a result, the Spanish WN containing 10,982 connections among 7,131synsets.

2.4.6. Building Polish WordNet (PolNet project)

The main goal in this paper [32] is to build a Polish WordNet from the scratch. They use a monolingual lexicon which has a well word senses and a semantic coverage.

The language independent algorithm is used to create the synsets. DebVisDic tool is used to create the relation between the synsets. WQuery tool is a system based on an artificial language designed to query WordNet as if it was a databases, and it is used to validate the PolNet data.

PolNet has 10,700 synsets. This result was obtained form 10,000 words that has been extracted from IPI PAN Corpus, Polish Lexicon of Verbs (Polański, 1992), and Corpora of text for the domain of homeland security terminology (1360 words), emergency situations dialogue corpus (630 words).

2.4.7. Building Czech WordNet

The core of Czech WordNet [33] has a good quality covering the most frequent Czech word and close as possible to the Princeton WordNet (WN) and the EuroWordNet (EWN), and it can be a good starting point for future testing and validation or building applications.

This method requires monolingual Czech dictionary, bilingual Czech-English dictionary, dictionary of Czech Synonyms, Czech Synonymic dictionary and Thesaurus I, II, III, fully tagged and disambiguated corpus DESAM, corpus from newspaper and magazine texts, Czech national corpus.

Developers use many tools and programs in order to build the Czech WordNet, such as, processing the dictionaries using a sorting program, analyze the dictionaries entries using a parsing program, process the bilingual dictionaries using translating program, a program able to compute mutual information, VisDic which is a WordNet editor and browser.

When building Czech synsets, some problems appear due to the translation to get the equivalent corresponding, which presents a gap between Czech and English. First problem would be the differences in lexical and concepts between Czech and English, which made it difficult to find their equivalents in Princeton WordNet but can find the English correspondence in general. Second problem is that some Czech synsets cannot find their equivalents in English at all, as there are some typologically and derivational morphology differences, such as, reflexive verbs, verb prefixation (single, double), diminutives (noun derivation by suffixation).

2.5. Background about evaluation methods that use similarity measures or distance measures

This section gives a background about some evaluation methods which use similarity or distance measures. Some of these methods will be used later in this thesis. One way of comparing distance measures is to study their retrieval performance in terms of precision and recall in a particular application area, such as content-based image retrieval where Euclidean distance is often used as a distance measure.

Similarity measures increase the system performance [28], it can be used in WordNet and thesaurus to find the similarity percentage [29] and to find the degree of lexical overlapping between the files which we want to compare [30]. Hence, understanding the distance measures approaches can be helpful to choose the proper approach for a particular application [42].

2.5.1. Precision and Recall

Precision and recall are common measures to evaluate information retrieval systems. They are based on the comparison between the expected and the relevant results which we get to evaluate system [34]. Before we go into more details let us determine some definitions to give a better understand for precision and recall [35].

True positive: it is the number of correct labeled items that belong to the selected class.

False positive: it is the numbers of incorrect labeled items, which belong to the selected class.

True negative: it is the number of items that are not labeled to the selected class but should be labeled.

Precision or confidence [36] measures the ratio of the correct correspondences (true positives) to the total number of all return correspondences [35] (true positives and false positives). Precision can be seen as a measure of exactness or quality.

Recall or sensitivity [36] measures the ratio of correct correspondences (true positives) to the total number of all expected correspondences that must be retrieved [35] (true positives and true negatives). Recall is a measurement of completeness.

Precision formula is given by:

$$P(A,R)= \frac{|P\cap A|}{|A|}$$

Recall is given by:

$$R(A,R) = \frac{|R\cap A|}{|R|}$$

Where R: is the relevant data and A: is the retrieved data.

Precision and recall depend on an understanding and measuring of relevant data. For some algorithm, high precision indicates that the retrieved results are more relevant than irrelevant one's, while high recall indicates the most retrieved results are relevant. Precision and recall have opposite direction, when precision increases recall decreases and vice versa [35].

2.5.2. F-Measure

We cannot consider F- measure without precision and recall; therefore, the F-measure formula is combined from the two metrics precision and recall.

F-measure or F-score [34] is used commonly to measure the standard performance measures in information retrieval especially in the tasks which the elements of class should be retrieved correctly without retrieving elements from other classes or when the relevant elements are rare. F-measure is also common in information extraction tasks such as the name entity recognition where most of the elements do not belong to a named class [37].

F-score reaches the best value at 1 and worst value at 0. Figure 2.12 shows the F-measure percentage value and its indication. When the percentage increases, the F-measure increases. Top value when its 100% which is equal to 1 [38].

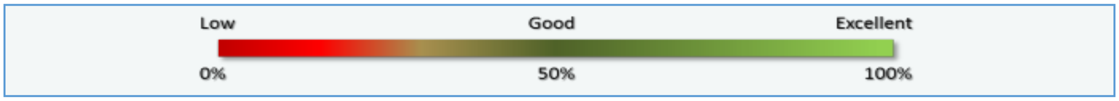


Figure 2.11: F-measure percentage value and its indication [38]

F-measure formula [39]:

$$F = \frac{(1+\beta^2) * recall * precision}{(\beta^{2*} precision) + \ recall}$$

ß controllers if we want to weight either precision or recall more heavily, then we choose the suitable ß number, when ß = 1, then precision and recall are balanced. In most experiments, the researchers use ß = 1 as there is no reason to give precision or recall more weight.

One way of comparing distance measures is to study their retrieval performance in terms of precision and recall [42].

2.5.3. Euclidean distance

Euclidean distance [40] calculates the distance which is used in geometrical problems such as clustering problems, including clustering text. It measures the distance between two points in two or three-dimensional space.

The Euclidean distance between two documents da and db is represented by their vectors ta and tb is defined as:

$$D_E(\overrightarrow{t_a}, \overrightarrow{t_b}) = (\sum_{t=1}^{m} |w_{t,a} - w_{t,b}|^2)^{1/2},$$

Where the term wt,a , wt,b are two points.

2.5.4. Cosine similarity

Similarity relation is a mathematical notion that provides a way to manage alternative instances of an entity that can be considered “equal” to other entities with a given degree [41].

Cosine similarity is a measure of similarity between two vectors in order to measure the cosine angle between them but not their magnitudes [42]. The cosine value can vary between 0 and 1, the bigger the value the more similar the two vector are[43].

Cosine similarity converts the string to a vector. This method uses a map data structure to represent string in vector to associate each word with its frequency value in the string [44].

In order to form a vector from two strings, firstly, for each string, we list the distinct words. Secondly, we count the frequency of each word in the list, the counter of each word is incremented by one each time it found the original string. These counters are then saved in an array. Thirdly, the process will be repeated for the second string. Finally, by formatting the two vectors, we can substitute these vectors in the cosine similarity equation that is represented below [44].

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

2.5.5. Jaccard coefficient

The Jaccard coefficient often used to compare the similarity, the dissimilarity, and distance between finite sets of objects by finding the intersection of objects divided by their union [47]:

$$J(A,B) = \frac{A \cap B}{A\ U\ B}$$

The formal definition is[40]:

$$S_{ij} = \frac{a}{a+b+c}$$

where, a is the number of attributes positive for both objects ,b is the number of attributes 1 for i and 0 for j and c is the number of attributes 1 for i and 0 for j. The Jaccard coefficient ranges between 0 and 1. It is 1 when the two objects are the same, and its 0 when are disjoint which means they are completely different.

2.6. Comparison between the evaluation methods

Our main goal of using an evaluation method is measure the similarity between two sets of words. A suitable evaluation method, for our research, should take into account small sets specially that a set in our research can be one to max 14 words.

Euclidean distance is dependent of the vector length. Vectors of different lengths will have a large Euclidean distance. Figure 2.13 shows a number of documents, the Euclidean distance between q and d2 is large even q and d2 are very similar and have a close distribution [45].

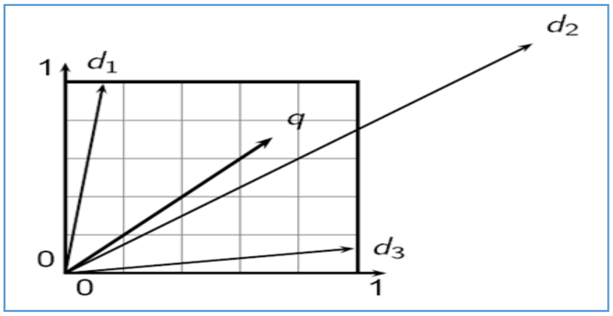


Figure 2.12: The Euclidean distance between *q* and d_2[45]

Cosine similarity is one of the most similarity measurements applied to text documents, as it is independent of the document length. Suppose we have a document d and its identical copy d′, the cosine similarity between d and d′ is 1, which means that these two documents are identical [40].

Jaccard similarity is particularly suited when dealing with data objects that have asymmetric binary attributes which defined as the number of common attributes is divided by the number of attributes that exists in at least one of the two objects. The most important question is to know if the extra bit of information can reflect or hurt or do nothing [46]. "*Jaccard and Cosine similarity both depends on common terms, but cosine similarity are weighted* " [48]. These weights are larger for words that are rare in the collection of data sets.

Jaccard similarity was used to build fuzzy thesaurus (Miyamoto, 1990; Ogawa et al., 1991), and Cosine similarity were used to build automatic thesaurus construction (Frakes & Yates, 1992) [46]. Both cosine similarity and Jaccard seem to be good choices to measure the distance between sets of words for our research.

Chapter 3

3. Automatic Generation of Synonyms

This chapter presents the SynsetGenerator algorithm [7] and its implementation. First, section 3.1 formalizes the problem and the goals we want to achieve. The algorithm will be presented and illustrated with a running example in section 3.2.Section 3.3 will present the implementation of the algorithm and technical issues faced during this research.

3.1. Formulation of the problem and goals

As mentioned earlier, the importance of the Arabic cross-lingual resources such as Arabic thesauruses and EWN, and Arabic ontologies is increasing rapidly, especially as they are being used as core components in ANLP such as text translating, text summarizing, and information retrieval.

The main objective of this research is to build synonyms automatically. In order to achieve this we divided the thesis into two phases. First phase is building Arabic thesaurus (i.e., sets of synonyms) automatically generated from existing Arabic bilingual dictionaries. The second phase is to evaluate the accuracy of generated synonyms by mapping them to WordNet synonyms. That is, given a set of synonyms $\{a_1, a_2, \ldots, a_n\} := \{e_1, e_2, \ldots, e_k\}$ that we generate and given WordNet synonyms, we will map both into each other i.e., $\{e_1, e_2, \ldots, e_n\} := \{wn_1, wn_2, \ldots, wn_m\}$, where a_n refers to an Arabic word, e_k refers to an English word, and wn_m is an WN's word. This part (mapping into WordNet) will be presented in chapter 4.

In the first phase, for a given bilingual dictionary (e.g. Arabic-English) we generate the possible Arabic synonyms for a certain Arabic word, for example [7]:

جدول: نهر، قناة ماء

جدول: مصفوفة، قائمة

Such sets of Arabic synonyms (called synset) are obtained by converting the bilingual dictionary into a graph where the first level is an Arabic word, the 2nd level is the English translations of the 1st level words, the 3rd level is the translation of 2nd Level, and so on, until no translation is found or a word is repeated (i.e., cycle).Then synonyms are the words in cyclic paths.

3.2. The SynsetGenerator algorithm

In this section we illustrate the steps followed to automatically generate all possible synonyms of this form: $\{a_1, a_2, \ldots, a_n\} := \{e_1, e_2, \ldots, e_k\}$. As mentioned earlier we decided to implement and use the SynsetGenerator Algorithm proposed by [7]. In the following we describe this algorithm, and in the next subsection we illustrate it using a running example.

Given a bilingual dictionary (i.e., one table with two columns Arabic Word and EnglishWord), the algorithm converts this dictionary into a graph, where each node is a word (which can be Arabic or English word). A link between words denotes translation of this word based on the given bilingual dictionary. The following steps demonstrate how the algorithm generates synonyms. Note that Lemmas of words are used in the algorithm instead of the words themselves.

Given a bilingual dictionary as a set of tuples of the form $<a_n,e_k>$, where a_n is word in a language (e.g., Arabic) and its translation into another language is e_k (e.g., English), do the following[7]:

Step-1: Lemmatize words in both languages. Each word a_i will be replaced with its Lemma la_i, and each e_j will be replaced with its le_j. This step is essential in order to avoid possible problems that might arise when comparing words against each other. As will be discussed later, there are several available Lemmatizes for most languages that one can use in this step.

Step-2: For each Arabic Lemma (la_i), find the corresponding English lemmas $\{le_1,le_2...le_i\}$, which are the translations found in the input bilingual dictionaries.

Sept-3: For each English Lemma {le_1… le_n} found in step-2, find the corresponding Arabic lemma, which are also the translations found in the input bilingual dictionary.

Step-4: Repeat **step-2** and **step-3** until either: (i) no translations are found, or until (ii) the resulted Arabic/English was generated earlier, or (iii) reach the Arabic lemma that we started from (i.e., a cycle detected). In other words, this step aims to build a graph of the possible translations by finding the translation of the Arabic lemma, then its English translations, then the Arabic translations of each English lemma, and so on. As described earlier, each lemma is seen as a node and the translation is seen as a link between two nodes. The algorithm stops either when no translations are found or a cycle is detected (i.e., when the retrieved lemma appeared in the path earlier).

Step-5: Convert all cyclic paths detected in step-3 into synonyms. If a path was found, for example ($\boldsymbol{a_1} \rightarrow e_1 \rightarrow a_2 \rightarrow e_2 \rightarrow a_3 \rightarrow e_5 \rightarrow a_7 \rightarrow e_9 \rightarrow \boldsymbol{a_1}$) where the loop was caused because the first and the last nodes are the same, it means that all words in this loop are synonyms, thus such loops can be converted into synonyms as $\{a_1, a_2, a_3, a_7\} := \{e_1, e_2, e_5, e_9\}$.

Step-6: Consolidate paths: this step is applied on all converted paths in step-4. That is, this step aims to consolidate the Arabic and English synsets such that:

- If two sets of synonymy in English are the same, we consolidate their Arabic equivalence, such as:

$$\{a_1, a_2, a_3\}:= \{e_1, e_2, e_3\}$$

$$\{a_3, a_5, a_6\}:= \{e_1, e_2, e_3\}$$

We merge the Arabic synsets to get one synset like:

$$\{e_1, e_2, e_3\} := \{a_1, a_2, a_3, a_5, a_6\}$$

- If two sets of synonymy in Arabic are the same, we consolidate their English equivalence, such as:

$$\{a_1, a_2, a_3\}:= \{e_1, e_2, e_3\}$$

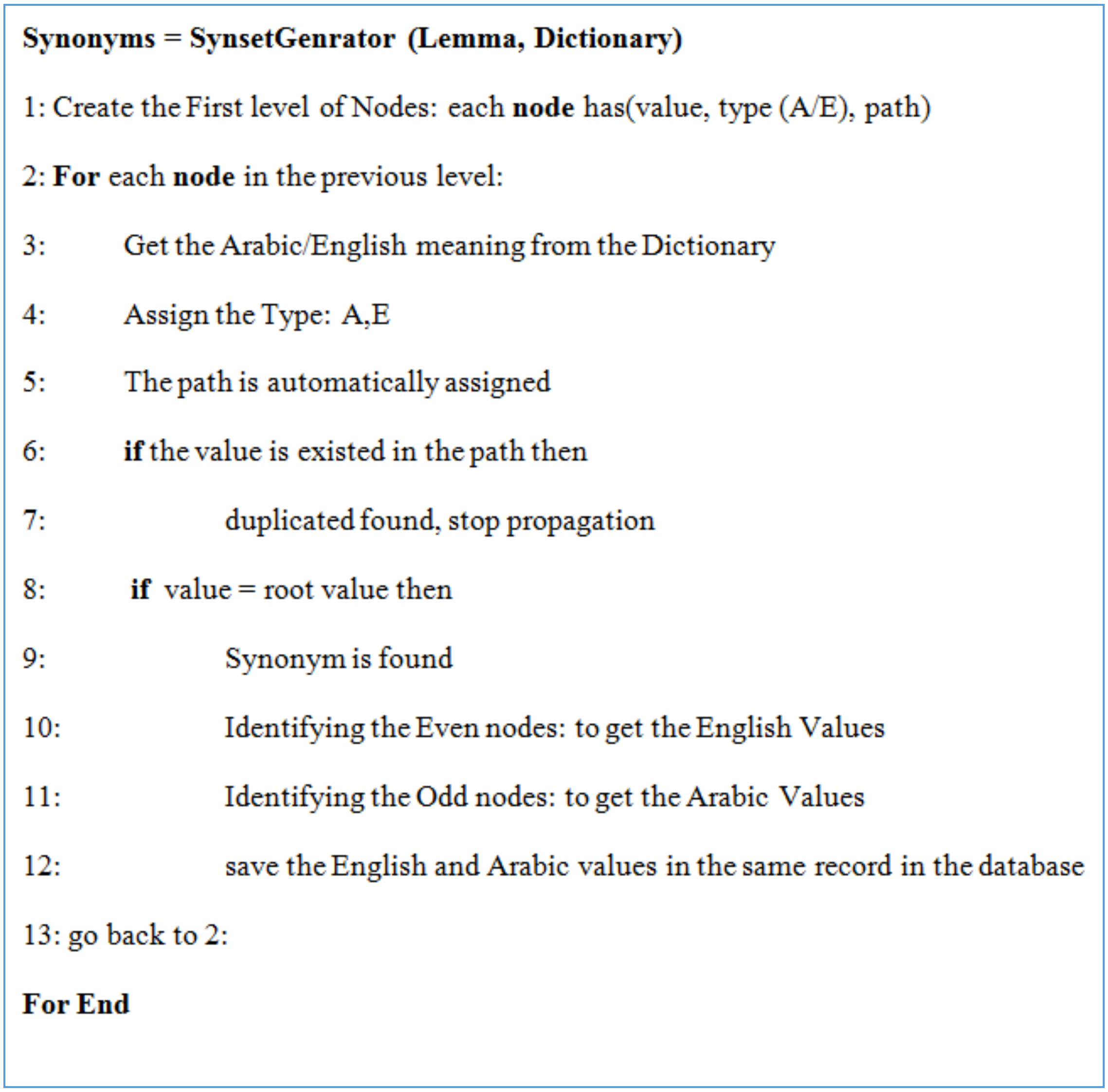
$$\{a_1, a_2, a_3\}:= \{e_4, e_5, e_6\}$$

We merge the English synsets to get one synset like:

$\{a_1, a_2, a_3\} := \{e_1, e_2, e_3, e_4, e_5, e_6\}$

- We repeat this step-6 until no consolidation in Arabic and English set of synonymy is needed; that is, we merge the paths generated in step-5 based on matches of sets of synonymy.

Figure 3.1 shows the pseudo code of the SynsetGenerator algorithm :

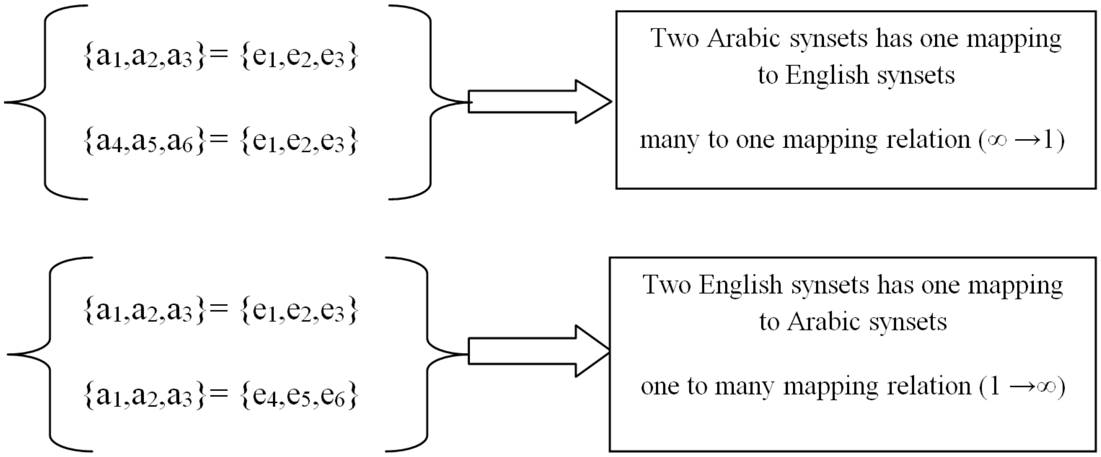


```
Synonyms = SynsetGenrator (Lemma, Dictionary)
1: Create the First level of Nodes: each node has(value, type (A/E), path)
2: For each node in the previous level:
3:      Get the Arabic/English meaning from the Dictionary
4:      Assign the Type: A,E
5:      The path is automatically assigned
6:      if the value is existed in the path then
7:          duplicated found, stop propagation
8:      if value = root value then
9:          Synonym is found
10:         Identifying the Even nodes: to get the English Values
11:         Identifying the Odd nodes: to get the Arabic Values
12:         save the English and Arabic values in the same record in the database
13: go back to 2:
For End
```

Figure 3.1: SynsetGenerator algorithm pseudo code

As we shall explain in details later in chapter four, our algorithm is designed with three assumptions in mind, which we suppose that they are true for any synset. The algorithm works ideally when they are true, if these assumptions are not met then the accuracy of our generated results might be affected:

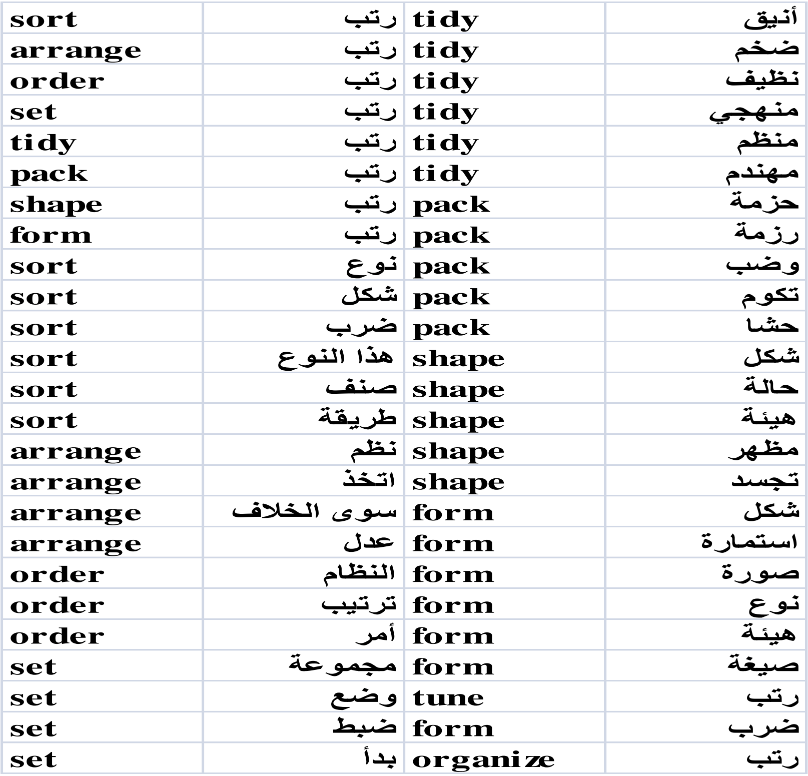
- Each synset is unique and has a unique synset ID.
- No synset can be a subset of another synset. For example: It's not allowed to have $synset_1$ has the following elements {A,B,C,D,E} and $synset_2$ has the following {A,B,C}.
- The linked AWN-ENW synsets are all one to one, i.e., there is only one English synset in EWN mapped to only one Arabic synset AWN, and vice versa. No different Arabic synsets have the same English synset or English synsets have the same Arabic synset, such that:



$\{a_1,a_2,a_3\} = \{e_1,e_2,e_3\}$
$\{a_4,a_5,a_6\} = \{e_1,e_2,e_3\}$
Two Arabic synsets has one mapping to English synsets
many to one mapping relation (∞ →1)
$\{a_1,a_2,a_3\} = \{e_1,e_2,e_3\}$
$\{a_1,a_2,a_3\} = \{e_4,e_5,e_6\}$
Two English synsets has one mapping to Arabic synsets
one to many mapping relation (1 →∞)

3.2.1. Example

This example is taken from [7]. It illustrates all steps of the SynsetGenerator algorithm explained above. Suppose we have the Arabic English bilingual dictionary presented in Figure 3.1, which is a table of two columns (English, Arabic) translations.



sort	رتب	tidy	أنيق
arrange	رتب	tidy	ضخم
order	رتب	tidy	نظيف
set	رتب	tidy	منهجي
tidy	رتب	tidy	منظم
pack	رتب	tidy	مهندم
shape	رتب	pack	حزمة
form	رتب	pack	رزمة
sort	نوع	pack	وضب
sort	شكل	pack	تكوم
sort	ضرب	pack	حشا
sort	هذا النوع	shape	شكل
sort	صنف	shape	حالة
sort	طريقة	shape	هيئة
arrange	نظم	shape	مظهر
arrange	اتخذ	shape	تجسد
arrange	سوى الخلاف	form	شكل
arrange	عدل	form	استمارة
order	النظام	form	صورة
order	ترتيب	form	نوع
order	أمر	form	هيئة
set	مجموعة	form	صيغة
set	وضع	tune	رتب
set	ضبط	form	ضرب
set	بدأ	organize	رتب

Figure 3.2: Arabic English bilingual dictionary[7]

Step-1: Lemmatizing of Arabic and English words. Here we find that the Arabic lemmas for the Arabic words are the same, and the English lemmas for the English words are the same.

Step-2: For each Arabic Lemma such as (رتب), find the corresponding English lemmas {sort, arrange, shape, order, clean, pack, tidy, set}, which are the translations found in the input bilingual dictionaries. Figure 3.2 shows the graph of the first level that resulted from the algorithm:

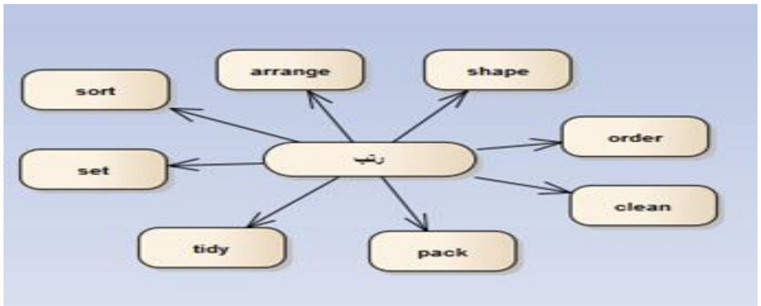


Figure 3.3: First level [7]

Sept-3: For each English Lemma such as {arrange}found in step-1, we find the corresponding Arabic lemmas, which are also the translations found in the input bilingual dictionary. Figure 3.3 shows the graph of the second level that resulted from the algorithm.

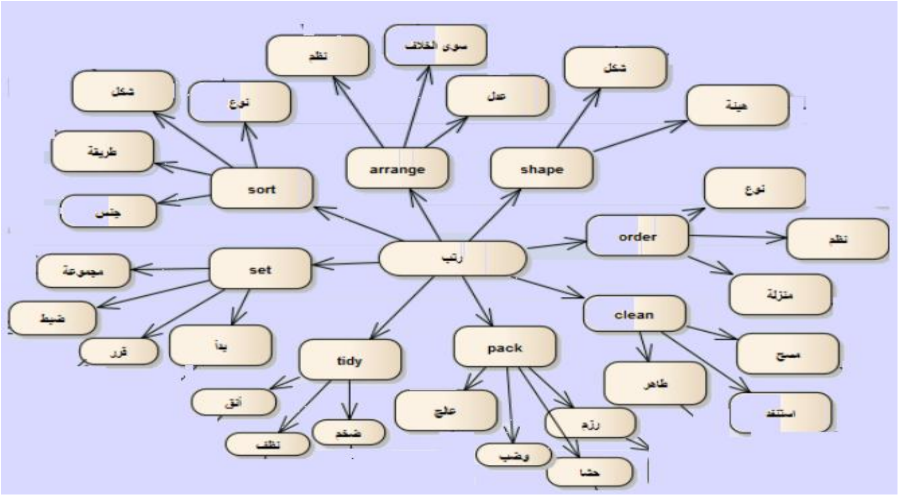


Figure 3.4: Second level [7]

Step-4: Repeat step-1 and step-2 until either (i) no translations are found, or until (ii) the resulted Arabic/English was generated earlier, or (iii) reach to the Arabic lemma which we started from.

Figure 3.4 shows the graph of the fourth level that will be resulted from the graph.

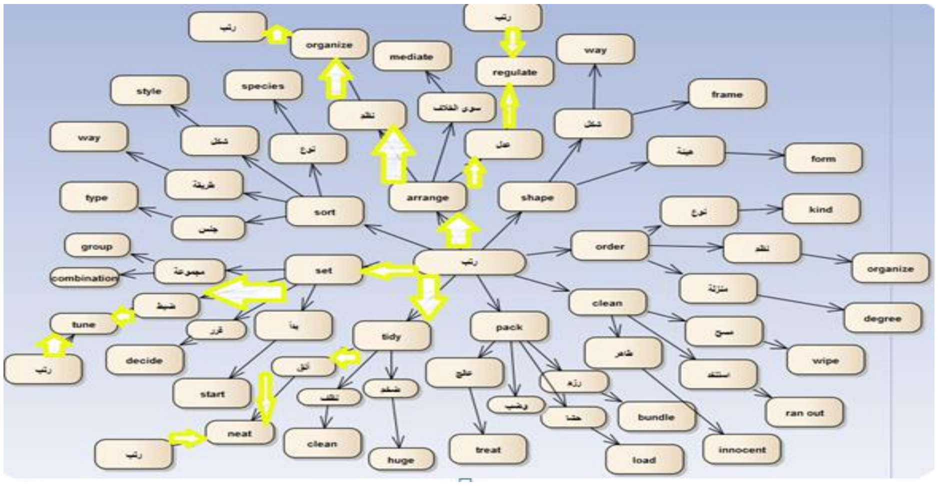


Figure 3.5: Fourth level [7]

Step-5: Convert all loops into synonyms. If a path was found, then the loops can be converted into synonyms. In figure 3.4 the yellow lines mark discovered paths. For example, each path started from the root node"رتب" and ends with the same word "رتب" presents a single path (*رتب→arrange→نظم→ organize→رتب*), which means that all words in this loop are synonyms, thus {رتب، نظم } :={arrange , organize} are synonyms . Figure 3.4 presents four paths for the word “رتب":

{رتب، نظم} := {arrange , organize}

{رتب، عدل} :={arrange، regulate}

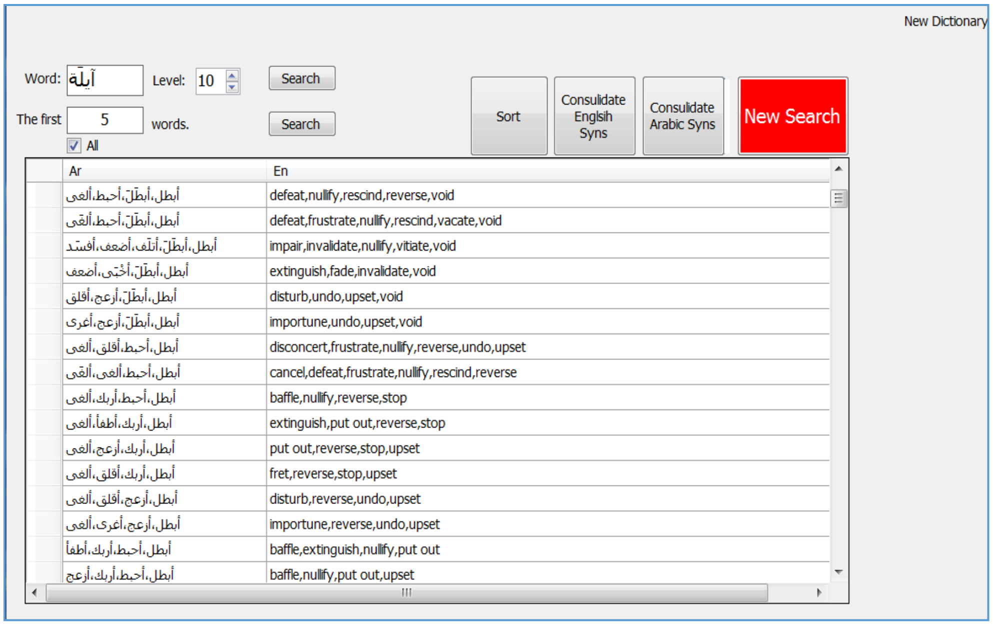
{ رتب،ضبط}:={set, tune}

{رتب، أنق} := {tidy، neat}

Step-6: Synonyms consolidation step is applied for all converted paths; In this example there are no consolidations needed as there are no two sets of synonymy in English or Arabic which are the same.

3.3. SynsetGenerator algorithm implementation

The previously presented algorithm has been fully implement and tested. Figure 3.4 shows the graphical interface to present results. Through this interface, we can generate the synonym set for a given Arabic word. In addition, we can generate synonymy for set of Arabic words by choosing the first number (*n*) of words as listed in the database. Moreover, we can generate synonymy for all Arabic words.



Word: آيلَة Level: 10

The first 5 words.

☑ All

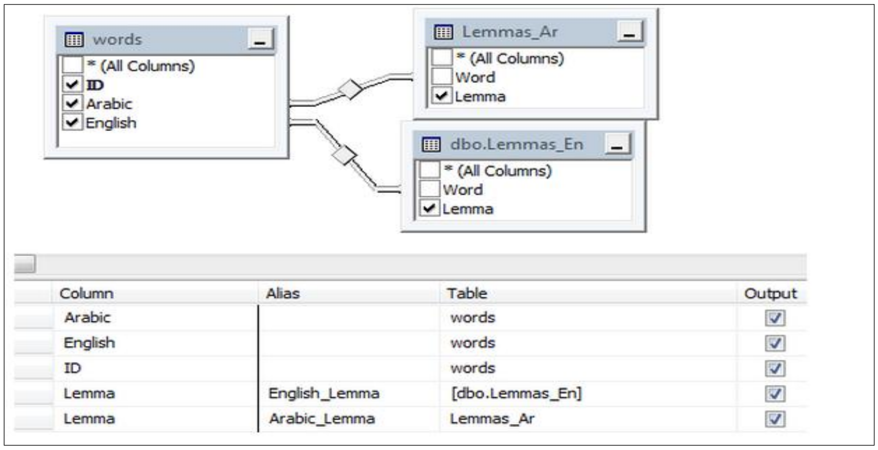
Ar	En
أبطل،أبطَلَ،أحبط،ألغى	defeat,nullify,rescind,reverse,void
أبطل،أبطَلَ،أحبط،ألقى	defeat,frustrate,nullify,rescind,vacate,void
أبطل،أبطَلَ،أتلف،أضعف،أفسَد	impair,invalidate,nullify,vitiate,void
أبطل،أبطَلَ،أخْبَى،أضعف	extinguish,fade,invalidate,void
أبطل،أبطَلَ،أزعج،أقلق	disturb,undo,upset,void
أبطل،أبطَلَ،أزعج،أغرى	importune,undo,upset,void
أبطل،أحبط،أقلق،ألغى	disconcert,frustrate,nullify,reverse,undo,upset
أبطل،أحبط،ألغى،ألقى	cancel,defeat,frustrate,nullify,rescind,reverse
أبطل،أحبط،أربك،ألغى	baffle,nullify,reverse,stop
أبطل،أربك،أطفأ،ألغى	extinguish,put out,reverse,stop
أبطل،أربك،أزعج،ألغى	put out,reverse,stop,upset
أبطل،أربك،أقلق،ألغى	fret,reverse,stop,upset
أبطل،أزعج،أقلق،ألغى	disturb,reverse,undo,upset
أبطل،أزعج،أغرى،ألغى	importune,reverse,undo,upset
أبطل،أحبط،أربك،أطفأ	baffle,extinguish,nullify,put out
أبطل،أحبط،أربك،أزعج	baffle,nullify,put out,upset

Figure 3.6: The interface of the SynsetGenerator algorithm

The algorithm default-state is to keep generating the nodes until no new nodes are added to the tree, however, if we want to choose a certain level, we can select the level number as shown in figure 3.4. The level number must be even (2,4,6,8, ...) as even level indicates that the path ends with Arabic word, odd level number indicates that the path ends with English word. The tree path stops when one of these conditions happens: (1) If we reach the Arabic lemma that we started with i.e., cycle is detected, (2) If we reach a node that is already in the path no matter this node is an Arabic or an English node i.e., cycle is detected, or (3) If we reach the wanted level, or no new nodes are added.

The algorithm uses the Arabic and English lemmas instead of the words themselves, so as a first step we need to **find the lemmas of the Arabic the English words.** For each of the Arabic and English words from Bilingual Arabic English dictionary, we need to find the lemma for these words, and then store them into a database. This is an important step as several words that have the same meaning (e.g., "إجازة", "الإجازة", and "الإجازات"), have the same lemma ("إجازة"), thus it is important to use the lemmas of the words instead of using the words themselves which will increase the algorithm accuracy, improves the performance and avoids repeating of the words within a path. In our implementation, we used MADA morphological analyzer to lemmatize our Arabic words and we used language tool desktop application to lemmatize the English words.

Figure 3.6 shows the **database schema** we have, which includes Words table, Arabic lemma Table, and English lemma Table.



words

□ * (All Columns)
☑ ID
☑ Arabic
☑ English

Lemmas_Ar

□ * (All Columns)
□ Word
☑ Lemma

dbo.Lemmas_En

□ * (All Columns)
□ Word
☑ Lemma

Column	Alias	Table	Output
Arabic		words	☑
English		words	☑
ID		words	☑
Lemma	English_Lemma	[dbo.Lemmas_En]	☑
Lemma	Arabic_Lemma	Lemmas_Ar	☑

Figure 3.7: Database schema for SynsetGenerator algorithm

The **second step** in the algorithm is to find the corresponding English lemmas for each Arabic Lemma, which are the translations found in the input bilingual dictionaries. The **third step** is to find the corresponding Arabic lemmas for each English lemma found in the first step, which are also the translations found in the input bilingual dictionaries. The **fourth step** of the algorithm is to repeat the first step and second steps until no new nodes are added to the tree or a node is repeated in the path.

The **fifth step** in the algorithm is to convert all loops into synonyms. When an Arabic node is created in the path, the algorithm will check whether this node represents a synonym where it checks if the first and the last node in the path are the same, then it divides the path nodes by two, i.e., separates the Arabic and English nodes into two sets, if the mod is equal to zero, then it's an Arabic node, else it is an English node. Once the algorithm determines they are synonyms, it will write the original words instead of their lemmas in the synsets which are written in the interface grad as shown in figure 3.5.

When the application builds all the synsets, these synsets are sorted through the sort button as shown in the interface (figure 3.5) to prepare the synsets for the next step. We sort the words of both Arabic and English sets.

L**ast step** of the algorithm performs synonyms consolidation where for the same English synset we find all Arabic synsets that corresponds to this English synset, for example:

{اطلق، أخرج، أمر}:= {discharge, eject, order}

{اطلق، أخرج، أدار}:= {discharge, eject, order}

Then we merge these Arabic synset to get one synset as shown below:

{اطلق، أخرج، أدار، امر}:= {discharge, eject, order}

This step is done through the *consolidate English synsets* button included in the interface. The same procedure is performed for Arabic synsets where if we have same Arabic synset for different English synsets, such as:

{أذى، اصاب،أهان ،الم}:= {infect, insult, inflict, insult}

{أذى، اصاب،أهان، الم}:= {hurt, inflect, insult, offend}

We merge the English synset to get one synset as shown below:

{أذى، اصاب،أهان، الم}:=   {infect, insult, inflict, insult , hurt, offend}

This step is done through the *consolidate Arabic synsets* button included in the interface. When all the above steps are applied **the automatic Arabic thesaurus** is created by exporting the resulted synsets into a text file. The resulted synset accuracy for of the **automatic Arabic thesaurus** (the output file) depends on the accuracy of the Bilingual dictionary (the input file).

The technology which we use to develop the algorithm is Visual Basic .Net which is connected to SQL server as a database. We import the excel sheet bilingual dictionary to the SQL Server database to be able to execute SQL statements.

Once we execute the application, all needed data are imported to a temporary location in the memory, this step guarantees a high speed execution of the select statements which is called ADO. Net (Active X Data Object).

The reason why we use .Net technology as it provides many controls and a good user interface GUI (Graphical user interface) which made the data presentation easier and more efficient. The main two controls which we use are:

1. The tree view : In our implementation, we use a tree structure control as it has a collection of ordered levels of nodes and each node is an object, and each object has a set of methods and properties. In our case we used the properties to give it the shape whether it is an Arabic node or an English one, and we use the methods to separate Arabic nodes apart from the English ones; also to create the paths that represent the synonym.

   The depth of the tree structure implies the number of levels which is far from the original node that we start from (the root node). The depth of the tree depends on the number of word synonyms. For example, if the tree depth is 10, then this implies that the root has 5 distinct Arabic synonyms and 5 distinct English synonyms.

   The tree hierarchy guarantees the processing of the nodes by the inner loops which help us to know:

   1. Whether the node is an Arabic node (if we have an even node) or English node (if we have an odd node).
   2. The value of the node.
   3. Comparing the node with the previous nodes in order to continue or stop the node propagation. The propagation is stopped when we reach the node which we created in a previous levels, or when we reach a node that is equal to the node we start from.

   After having all the nodes processed through the inner loops, the output will be reflected into a data grid.

2. The data grid view: in our implementation, we also use a data grid view which is a table. This table let the user see the synonyms creation process which applied in user interface.

   After finishing the synonyms creation process, .Net supports the exporting to a format such as xml format for the data backup. That is, if the application is suddenly stopped without finishing all synonyms creation, then the next execution of the application will continue from the point which was stopped from, we also use the txt format to export the final synonyms.

Visual basic .Net has many built in functions which we use, such as the split function to know the number of words inside each path.

Chapter 4

4. Evaluation

In this chapter we evaluate the algorithm accuracy, mainly, the generated sets of synonyms, which is considered the second core part of our research. Sections 4.1 and 4.2 explain the experiment idea and describe the steps followed to **test the SynsetGenerator algorithm** implementation. We describe the steps followed to test some assumptions in section 4.3. Section 4.4 presents mapping algorithm implementation used to map the Arabic synsets that resulted from the **automatic Arabic thesaurus** algorithm to the EWN concepts, and the percentage of the mapped synsets. Finally, section 4.5 presents the methodology through which we can use the mapping algorithm to find the number and percentage of exact WN synsets that are generated by the **SynsetGenerator algorithm** compared to the original linked WN synsets.

4.1. Experiment idea

The correctness of synonyms generated by our implementation clearly depends on the correctness of the input bilingual dictionary i.e., if the bilingual dictionary we used in the initial phase contains incorrect or weak translations, then our algorithm will generate incorrect results. Thus, the better the quality of translations in the input dictionary, the better the quality of our results. To test this claim, we conducted the following experiment: We took the Arabic WordNet (contains Arabic synsets mapped with their equivalent synsets in the English WordNet) and extracted an Arabic–English bilingual dictionary from it. Then we used our implementation to generate Arabic and English synsets, and then we compared these generated synsets with the original synsets. Our evolution will be: How much our implementation is able to generate the same original synsets. We shall use cosine similarity as a mathematical tool to compare synsets.

4.2. Experiment setup

This section describes the steps needed to conduct the evaluation. First we downloaded both the Arabic WordNet and the English WordNet (version 3.1), with their Arabic-English mappings. Then we stored them in a relational database, the number of AWN is 10426 synsets and the EWN is 117791 synsets. We then performed the following step:

Step-1: Generate the Arabic and English linked synsets by developing WN synset application. The result of the application will be presented as: $\{a_1, a_2, a_3\} := \{e_1, e_2\}$ where **a** is an Arabic word from AWN, and **e** is an English word from EWN.

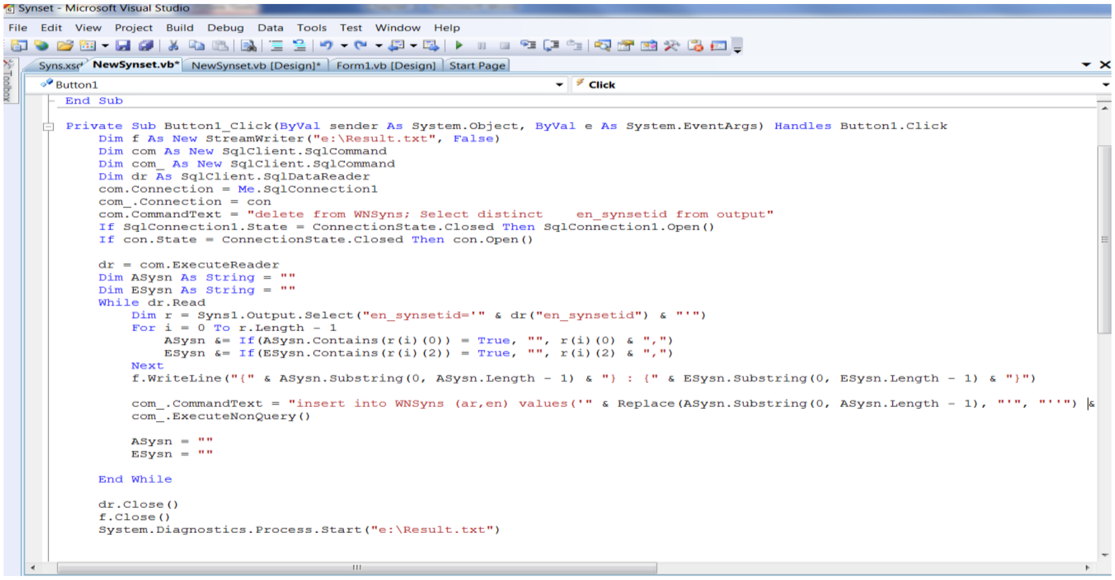
Step-2: Convert the generated linked synsets into output table with three fields(Ar, Syns ID, En). The conversion process was done by joining the tables in step-1 by Syns ID. The output table entries that are presented as $\boldsymbol{\{a_1, a_2, a_3\} := \{e_1, e_2\}}$ will be converted into $\{a_1,e_1\}$, $\{a_1, e_2\}$, $\{a_2, e_1\}$, $\{a_2, e_2\}$, $\{a_3, e_1\}$, $\{a_3, e_2\}$ sets.

Step-3: Connect the output table to **SynsetGenerator** implemented algorithm as an input, and generate the WN synsets again $\{a_1, a_2, a_3\} := \{e_1, e_2\}$ as output.

Step-4: Compare the results from step 1 and step3, If same synsets are produced, then the program implementation draws the graph correctly.

4.2.1. Testing our implementation

Figure 4.1 shows our application, which generates linked synsets from both AWN and EWN.

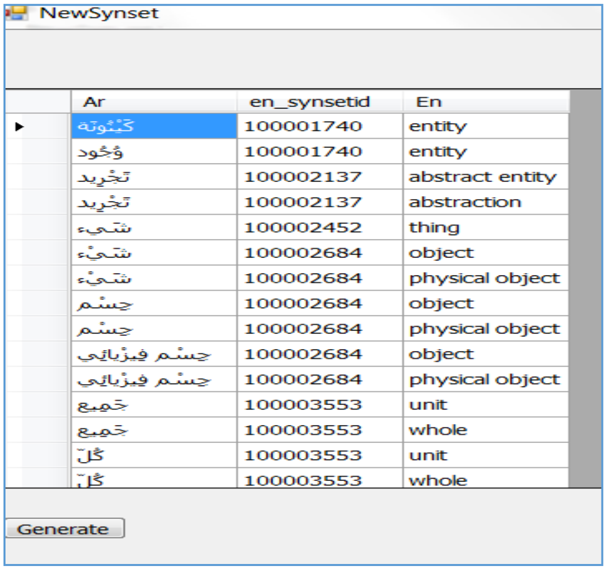


Synset - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Test Window Help
Syns.xsd NewSynset.vb* NewSynset.vb [Design]* Form1.vb [Design] Start Page
Button1
Click

```
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim f As New StreamWriter("e:\Result.txt", False)
    Dim com As New SqlClient.SqlCommand
    Dim com_ As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    com.Connection = Me.SqlConnection1
    com_.Connection = con
    com.CommandText = "delete from WNSyns; Select distinct    en_synsetid from output"
    If SqlConnection1.State = ConnectionState.Closed Then SqlConnection1.Open()
    If con.State = ConnectionState.Closed Then con.Open()

    dr = com.ExecuteReader
    Dim ASysn As String = ""
    Dim ESysn As String = ""
    While dr.Read
        Dim r = Syns1.Output.Select("en_synsetid='" & dr("en_synsetid") & "'")
        For i = 0 To r.Length - 1
            ASysn &= If(ASysn.Contains(r(i)(0)) = True, "", r(i)(0) & ",")
            ESysn &= If(ESysn.Contains(r(i)(2)) = True, "", r(i)(2) & ",")
        Next
        f.WriteLine("{" & ASysn.Substring(0, ASysn.Length - 1) & "} : {" & ESysn.Substring(0, ESysn.Length - 1) & "}")

        com_.CommandText = "insert into WNSyns (ar,en) values('" & Replace(ASysn.Substring(0, ASysn.Length - 1), "'", "''") &
        com_.ExecuteNonQuery()

        ASysn = ""
        ESysn = ""

    End While

    dr.Close()
    f.Close()
    System.Diagnostics.Process.Start("e:\Result.txt")
```

Figure 4.1: WN synset application code

In this application we merge the WN's tables to get one table as an output that has 3 fields (Ar, SynsID, En) which build the link synset within the en_synsetid field. In this step we find that there is no need to convert the linked synsets $\{\mathbf{a_1, a_2, a_3}\}\mathbf{:=}\{\mathbf{e_1,e_2}\}$ to $\{a_1,e_1\},\{a_1,e_2\},\{a_2,e_1\}, \{a_2,e_2\},\{a_3,e_1\},\{a_3,e_2\}$, as this step is already done by merging the WN tables. When the application is executed a table will appear as shown in figure 4.2:



Ar	en_synsetid	En
كَيْنُونة	100001740	entity
وُجُود	100001740	entity
تجْرِيد	100002137	abstract entity
تجْرِيد	100002137	abstraction
شَيء	100002452	thing
شَيْء	100002684	object
شَيْء	100002684	physical object
جِسْم	100002684	object
جِسْم	100002684	physical object
جِسْم فِيزْيائِي	100002684	object
جِسْم فِيزْيائِي	100002684	physical object
جَمِيع	100003553	unit
جَمِيع	100003553	whole
كُلّ	100003553	unit
كُلّ	100003553	whole

Figure 4.2: The interface of the WN synset application

Once we click on *Generate* button a text file with a linked synsets from both WNs tables is generated which contains 8,799 synsets, with distinct synsets of 8,660. After that, we connect the output table to **SynsetGenerator** implemented algorithm as input trying to generate the WN synsets {a_1, a_2, a_3} := {e_1, e_2} as output again. By comparing the synsets, which are generated by linked WN Synset application in the testing part and WN synsets, which are generated by **SynsetGenerator** application, we obtained the results shown below:

- The **SynsetGenerator** built(**10752 generated synsets**). This number is greater than the WN synsets; which means that more synsets were created generated.
- These synsets will be evaluated to find the percentage of the exact match synsets using the mapping algorithm(section 4.4.1).

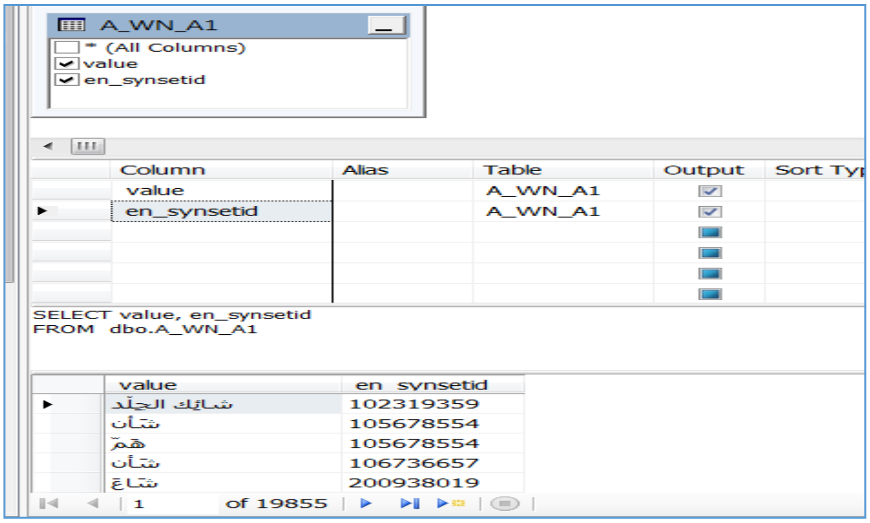
4.3. Testing WN assumptions

In order to find the reasons that caused our algorithm to not create the other synsets, we test the assumptions which we suppose about the WordNet in the next subsections.

4.3.1. Assumption one

Is each set of synonyms grouped into a synset is given a unique ID or there are same sets of synonyms given different IDs, in each of AWN and WN? Does a synset in AWN and EWN has a unique synset ID?

In order to check this assumption for both AWN and EWN, a Synset application is built. This application is applied once for AWN to build the AWN synsets, and then applied to EWN to build the EWN synsets. The application functions by putting all the words that have the same SynsetID together into the same synset, after that it puts the SynsetID beside the Synset. Figure 4.5 shows the A_WN_A1 table, which stores the AWN words and their SynsetID, and figure 4.6 shows the view which stores the EWN words and their SynsetID. This view has two tables; the first table is the E-word which contains the word and the wordID, the second table E_word_syns_ids which contains the word id and the synset id. This view joins the two tables to get the word and its SynsetID.



A_WN_A1

□ * (All Columns)
☑ value
☑ en_synsetid

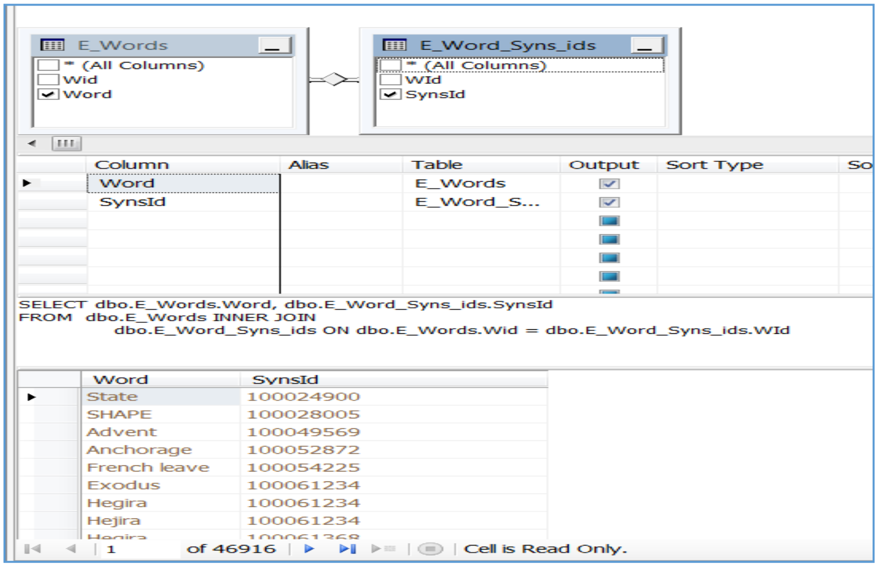
Column	Alias	Table	Output	Sort Typ
value		A_WN_A1	☑	
en_synsetid		A_WN_A1	☑	

```
SELECT value, en_synsetid
FROM   dbo.A_WN_A1
```

value	en synsetid
شائك الجلْد	102319359
شَأن	105678554
هَمّ	105678554
شَأن	106736657
شَاع	200938019

1 of 19855

Figure 4.3: AWN words table



E_Words

□ * (All Columns)
□ Wid
☑ Word

E_Word_Syns_ids

□ * (All Columns)
□ WId
☑ SynsId

Column	Alias	Table	Output	Sort Type	So
Word		E_Words	☑		
SynsId		E_Word_S...	☑		

```
SELECT dbo.E_Words.Word, dbo.E_Word_Syns_ids.SynsId
FROM   dbo.E_Words INNER JOIN
              dbo.E_Word_Syns_ids ON dbo.E_Words.Wid = dbo.E_Word_Syns_ids.WId
```

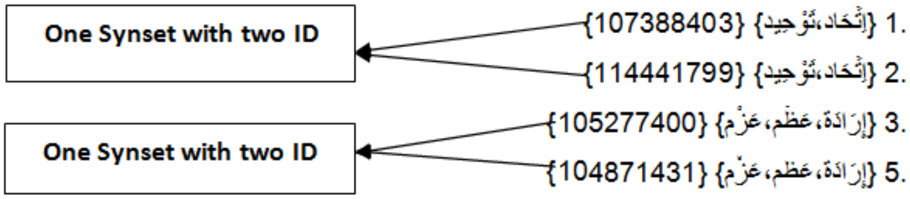
Word	SynsId
State	100024900
SHAPE	100028005
Advent	100049569
Anchorage	100052872
French leave	100054225
Exodus	100061234
Hegira	100061234
Hejira	100061234
Hegira	100061368

1 of 46916 | Cell is Read Only.

Figure 4.4: EWN words view

AWN Synset application builds the Arabic synsets, and EWN Synset application builds the English synsets. These applications write the results to a text file. The number of AWN synsets was 10426 synsets and the EWN synsets are 117791 synsets, after the synsets are built, the results are put in an excel sheet to find if we have a synsets with two SynsetIDs. The results obtained are shown below:

For the AWN: 1528 synsets from10426 synsets have more than one SynsetID, which is about 15% of the synsets. See appendix 1, and here is an example of the result obtained is shown below:


One Synset with two ID
{107388403} {إتِّحاد،تَوْحِيد} 1.
{114441799} {إتِّحاد،تَوْحِيد} 2.
One Synset with two ID
{105277400} {إرَادَة، عَظم، عَزْم} 3.
{104871431} {إرَادَة، عَظم، عَزْم} 5.

The synset {إِتِّحَاد،تَوْحِيد} has two synset IDs where each ID has a different gloss:

- {إِتِّحَاد،تَوْحِيد} {107388403}: =an occurrence that results in things being united.
- {إِتِّحَاد،تَوْحِيد} {114441799}:= there is strength in union.

has two synset IDs where each ID has a different gloss:{إِرَادَة،عَظْم،عَزْم}The synset

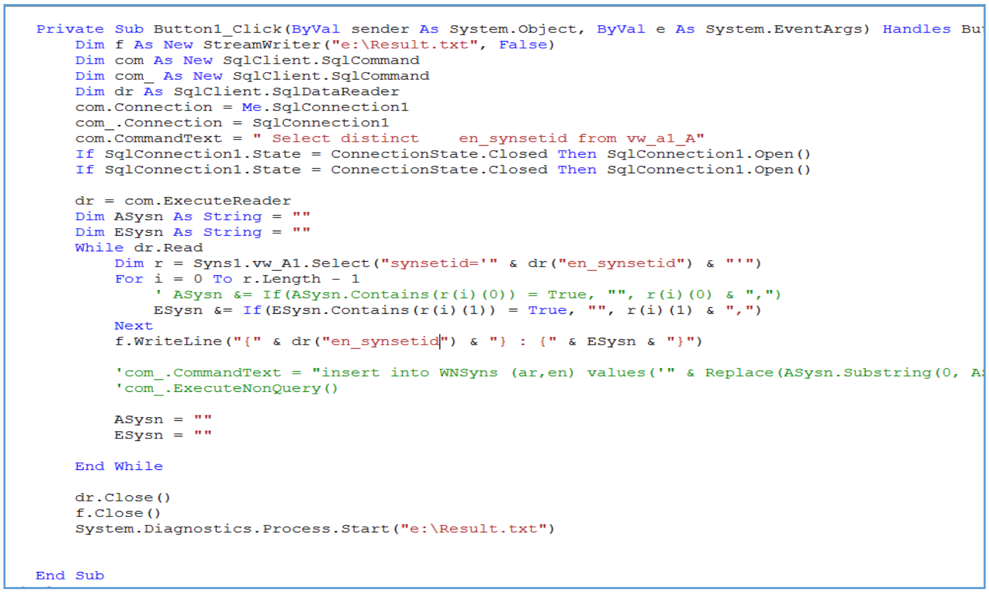
- {إِرَادَة،عَظْم،عَزْم} {105277400}:= Rigid connective tissue that makes up the skeleton of vertebrates.
- {إِرَادَة،عَظْم،عَزْم}{104871431}:= The trait of resoluteness as evidenced by firmness of character or purpose.

The table 4.1 shows the number of AWN synsets that have more than one SynsetID and contain one word, two words, three words and more than three words.

Num	**Num of words in AWN synsets that has more than one SynsetID**	**Num of AWN synsets that has more than one SynsetID**
1.	One word / Arabic synset	416
2.	Two words / Arabic synset	736
3.	Three words / Arabic synset	272
4.	Four words/ Arabic synset	84
5.	More than four words/ Arabic synset	20

Table 4.1: The number of words in AWN synsets that have more than one SynsetID

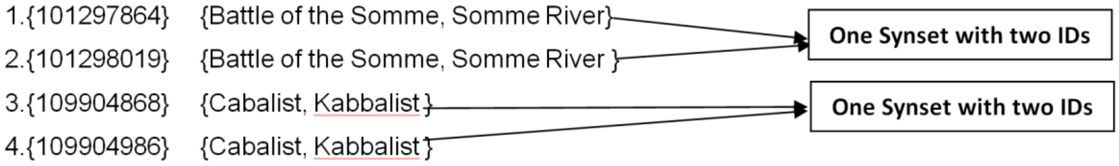
Figure 4.7 shows the AWN synset application, which is used to build the AWN synsets. Figure 4.8 shows the execution of AWN application. The application is also used to build the EWN synsets but it uses the EWN data to do so.



```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bu
    Dim f As New StreamWriter("e:\Result.txt", False)
    Dim com As New SqlClient.SqlCommand
    Dim com_ As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    com.Connection = Me.SqlConnection1
    com_.Connection = SqlConnection1
    com.CommandText = " Select distinct    en_synsetid from vw_a1_A"
    If SqlConnection1.State = ConnectionState.Closed Then SqlConnection1.Open()
    If SqlConnection1.State = ConnectionState.Closed Then SqlConnection1.Open()

    dr = com.ExecuteReader
    Dim ASysn As String = ""
    Dim ESysn As String = ""
    While dr.Read
        Dim r = Syns1.vw_A1.Select("synsetid='" & dr("en_synsetid") & "'")
        For i = 0 To r.Length - 1
            ' ASysn &= If(ASysn.Contains(r(i)(0)) = True, "", r(i)(0) & ",")
            ESysn &= If(ESysn.Contains(r(i)(1)) = True, "", r(i)(1) & ",")
        Next
        f.WriteLine("{" & dr("en_synsetid") & "} : {" & ESysn & "}")

        'com_.CommandText = "insert into WNSyns (ar,en) values('" & Replace(ASysn.Substring(0, A
        'com_.ExecuteNonQuery()

        ASysn = ""
        ESysn = ""

    End While

    dr.Close()
    f.Close()
    System.Diagnostics.Process.Start("e:\Result.txt")


End Sub
```

Figure 4.5: AWN Synset application

For the EWN: 42284 synsets over 117791 synsets has more than one SynsetID which is about 13 % of the synsets. See appendix 2, here is an example of the result obtained:



1.{101297864} {Battle of the Somme, Somme River}
2.{101298019} {Battle of the Somme, Somme River }
3.{109904868} {Cabalist, Kabbalist}
4.{109904986} {Cabalist, Kabbalist}

The synset {Battle of the Somme, Somme River} has two synset IDs where each ID has a different gloss:

- {Battle of the Somme, Somme River} {101297864}**:** =Battle in World War I (1916)**.**
- {Battle of the Somme, Somme River}{101298019}**:** =Battle of World War II (1944)**.**

The synset {Cabalist, Kabbalist} has two synset ID where each ID has a different gloss:

- {Cabalist, Kabbalist,} {109904868}:= A student of the Jewish Kabbalah.
- {Cabalist, Kabbalist,} {109904986}:= An expert who is highly skilled in obscure, difficult, or esoteric matters.

The table below shows the number of EWN synsets that have more than one SynsetID and contain one word, two words, three words and more than three words.

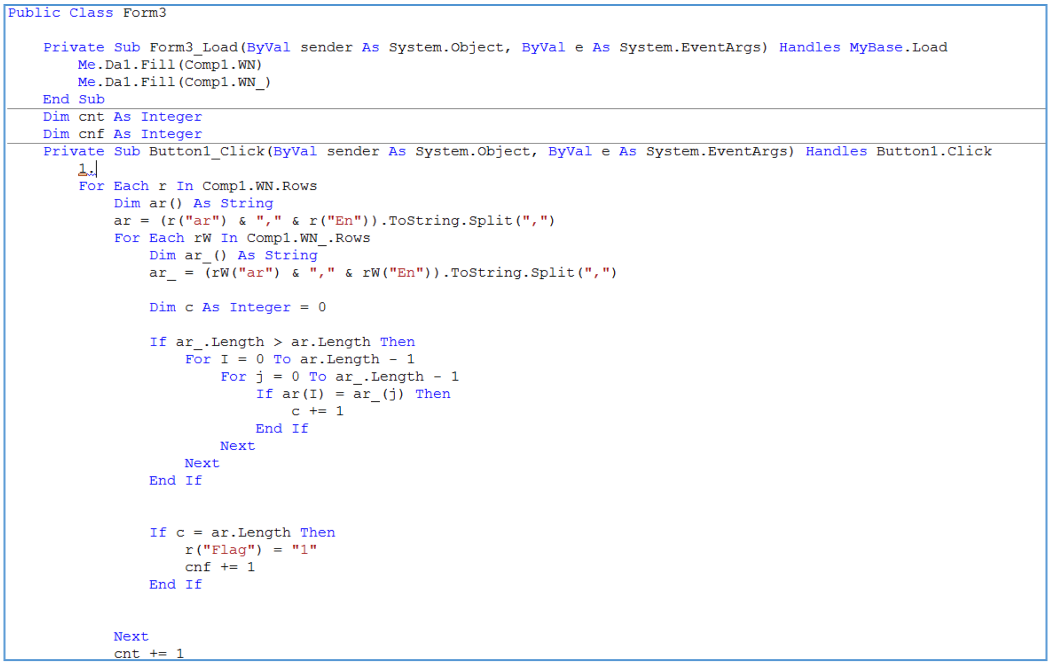
Num	**No. of words in EWN synsets that has more than one SynsetID**	**No. of EWN synsets that has more than one SynsetID**
1.	One word / English synset	27418
2.	Two words / English synset	11660
3.	Three words / English synset	2385
4.	Four words / English synset	615
5.	More than four words/ English synset	209

Table 4.2: The number of words in EWN synsets that have more than one SynsetID

4.3.2. Assumption two

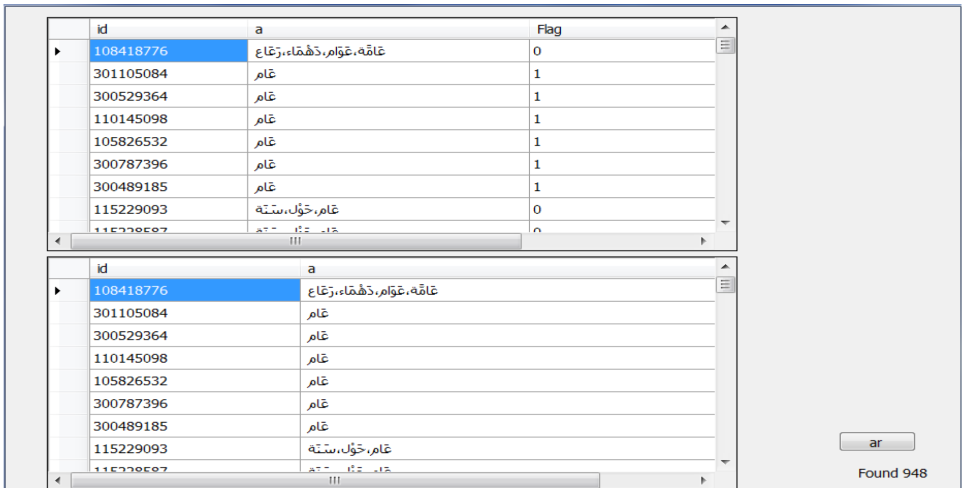
Are there any subsets in the AWN and EWN such that, synset1 has {A, B, C, D, E} elements and synset2 has {A, B, C} elements?

In order to check this assumption for both AWN and EWN separately, a compare synsets application is built. This application is applied to the AWN synsets that resulted from AWN synset application and EWN synsets that resulted from EWN synset application in the previous assumption. We compare each AWN Synset with all AWN Synset, then the application counts the number of subsets in the AWN. The same is done for EWN synsets. Figure 4.9 shows the Compare Synset application. Figure 4.10 shows the execution of Compare AWN Synset application. And figure 4.11 shows the execution of Compare EWN Synset application.



```
Public Class Form3

    Private Sub Form3_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Da1.Fill(Comp1.WN)
        Me.Da1.Fill(Comp1.WN_)
    End Sub
    Dim cnt As Integer
    Dim cnf As Integer
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        1.
        For Each r In Comp1.WN.Rows
            Dim ar() As String
            ar = (r("ar") & "," & r("En")).ToString.Split(",")
            For Each rW In Comp1.WN_.Rows
                Dim ar_() As String
                ar_ = (rW("ar") & "," & rW("En")).ToString.Split(",")

                Dim c As Integer = 0

                If ar_.Length > ar.Length Then
                    For I = 0 To ar.Length - 1
                        For j = 0 To ar_.Length - 1
                            If ar(I) = ar_(j) Then
                                c += 1
                            End If
                        Next
                    Next
                End If


                If c = ar.Length Then
                    r("Flag") = "1"
                    cnf += 1
                End If


            Next
            cnt += 1
```

Figure 4.6: Compare synset application



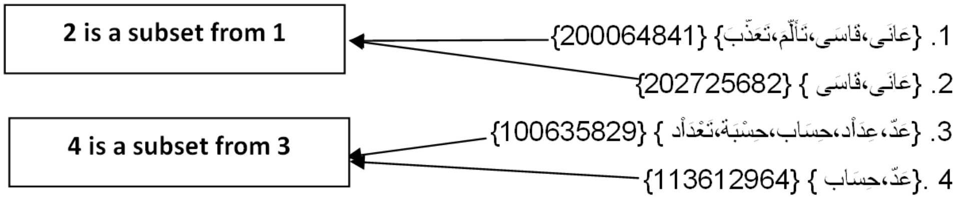
id	a	Flag
108418776	عَاقّة،عَوّام،دَهْمَاء،رَعَاع	0
301105084	عَام	1
300529364	عَام	1
110145098	عَام	1
105826532	عَام	1
300787396	عَام	1
300489185	عَام	1
115229093	عَام،حَوْل،سَنَة	0

id	a
108418776	عَاقّة،عَوّام،دَهْمَاء،رَعَاع
301105084	عَام
300529364	عَام
110145098	عَام
105826532	عَام
300787396	عَام
300489185	عَام
115229093	عَام،حَوْل،سَنَة

Found 948

Figure 4.7: The execution of compare AWN synset application

The results of the Compare AWN synset application: 948 synsets of the 10426 synsets are subsets of other synsets, which is about 9% of the synsets. See appendix 3, here is an example of the result obtained is shown below.



2 is a subset from 1

4 is a subset from 3

1. {عانَى،قاسَى،تَألّمَ،تعَذّبَ} {200064841}

2. {عانَى،قاسَى } {202725682}

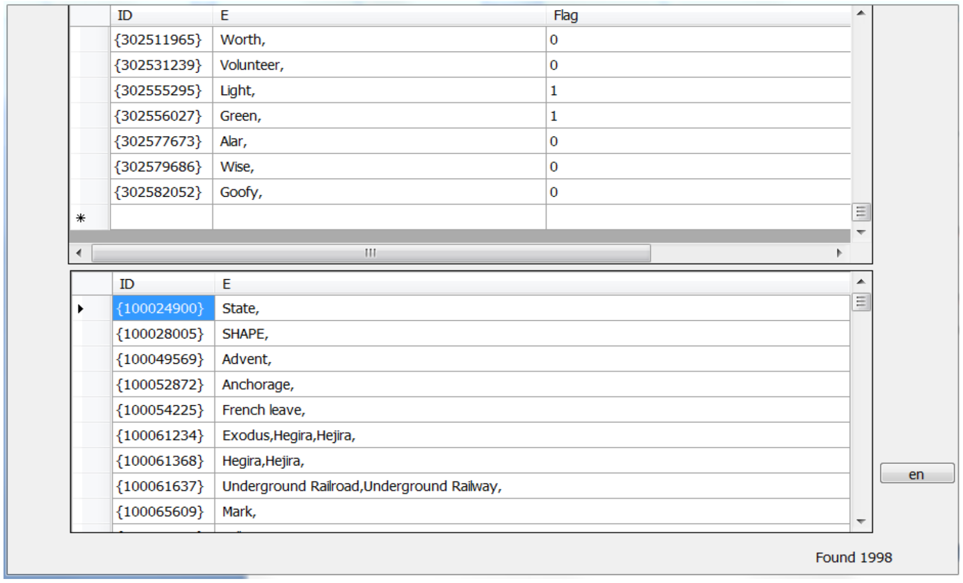
3. {عَدّ،عِدَاْد،حِسَاب،حِسْبَة،تَعْدَاْد } {100635829}

4. {عَدّ،حِسَاب } {113612964}

The table 4.3 shows the number of AWN synsets that have subsets and contain one word, two words, three words and more than three words.

Num	**Num of words in AWN synsets that has subsets**	**Num of AWN synsets that has subsets**
1.	One word / Arabic synset	756
2.	Two words / Arabic synset	155
3.	Three words / Arabic synset	32
4.	Four words / Arabic synset	4
5.	More than Four words/ Arabic synset	1

Table 4.3: The number of words in AWN synsets that have subsets



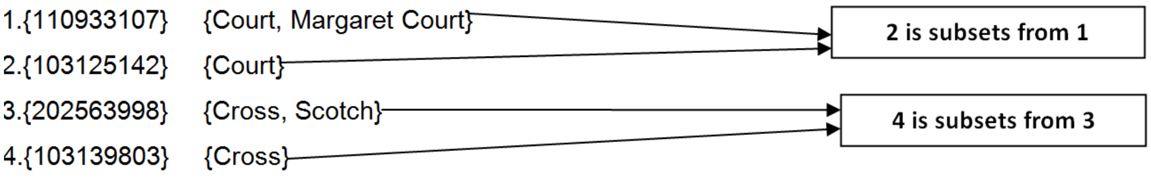
ID	E	Flag
{302511965}	Worth,	0
{302531239}	Volunteer,	0
{302555295}	Light,	1
{302556027}	Green,	1
{302577673}	Alar,	0
{302579686}	Wise,	0
{302582052}	Goofy,	0

ID	E
{100024900}	State,
{100028005}	SHAPE,
{100049569}	Advent,
{100052872}	Anchorage,
{100054225}	French leave,
{100061234}	Exodus,Hegira,Hejira,
{100061368}	Hegira,Hejira,
{100061637}	Underground Railroad,Underground Railway,
{100065609}	Mark,

Found 1998

Figure 4.8: The execution of Compare EWN Synset application

The results of the Compare EWN synset application:1,998 synsets of 117791synsets are subsets, which constitute about 2% of the synsets. See appendix 4, here is an example of the result is shown below.



1.{110933107} {Court, Margaret Court}
2.{103125142} {Court}
3.{202563998} {Cross, Scotch}
4.{103139803} {Cross}

The table below shows the number of EWN synsets that have subsets and contain one word, two words, three words and more than three words.

Num	Num of words in EWN synsets that has subsets	Num of EWN synsets that has subsets
1.	One word / English synset	1880
2.	two words / English synset	109
3.	three words / English synset	6
4.	Four words / English synset	1
5.	More than four word /English synset	2

Table 4.4: The number of words in EWN synsets that have subsets

4.3.3. Assumption three

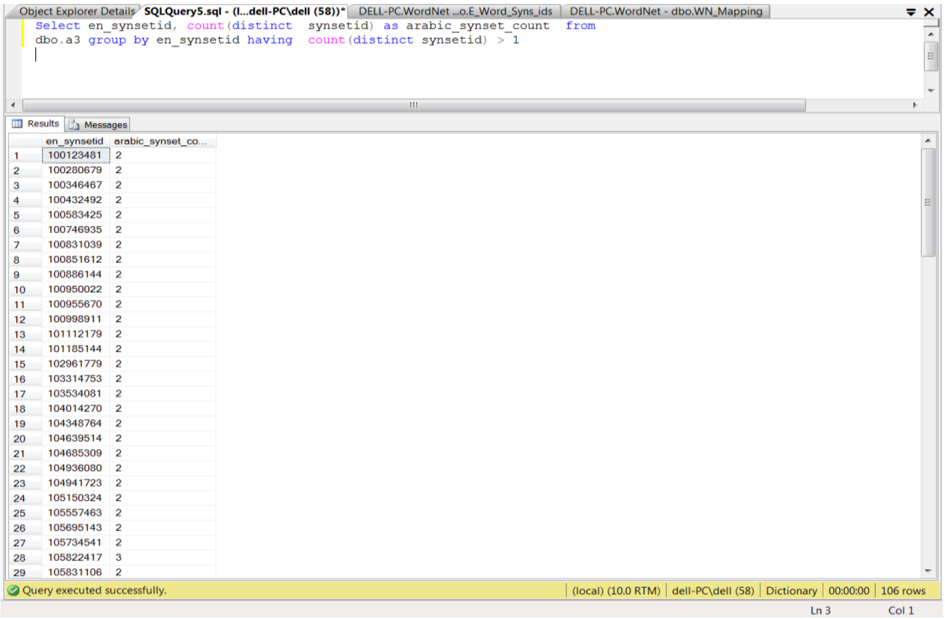
Is the relation one-to-one between the linked AWN and EWN synsets? Is each English Synset mapped to one Arabic synset and vice versa?

In order to check this assumption for both AWN and EWN, we use the result file from the Synset application, which builds the linked AWN and EWN synsets. We found that not all the 8,799 WN linked synsets has a relation of one to one. Some synsets have one to many relations (1 →n) such as:

- For the linked synsets: 106 synsets out of 8,799 have one English Synset which is mapped to many Arabic synsets. This means that the relation is (1 →n) and these synsets constitute about 1.2% of the synsets. Figure 4.12 shows a screenshot which applied mapped synsets to get the result (1 →n) .

The query used to find the number of English synsets that are mapped to many Arabic synsets is the following:

```
Select en_synsetid, count(distinct synsetid) as arabic_synset_count from
dbo.a3 group by en_synsetid having count(distinct synsetid) > 1
```



Object Explorer Details
SQLQuery5.sql - (l...dell-PC\dell (58))*
DELL-PC.WordNet ...o.E_Word_Syns_ids
DELL-PC.WordNet - dbo.WN_Mapping
Results
Messages

```sql
Select en_synsetid, count(distinct  synsetid) as arabic_synset_count  from
dbo.a3 group by en_synsetid having  count(distinct synsetid) > 1
```

	en_synsetid	arabic_synset_co...
1	100123481	2
2	100280679	2
3	100346467	2
4	100432492	2
5	100583425	2
6	100746935	2
7	100831039	2
8	100851612	2
9	100886144	2
10	100950022	2
11	100955670	2
12	100998911	2
13	101112179	2
14	101185144	2
15	102961779	2
16	103314753	2
17	103534081	2
18	104014270	2
19	104348764	2
20	104639514	2
21	104685309	2
22	104936080	2
23	104941723	2
24	105150324	2
25	105557463	2
26	105695143	2
27	105734541	2
28	105822417	3
29	105831106	2

```
Query executed successfully.    (local) (10.0 RTM) | dell-PC\dell (58) | Dictionary | 00:00:00 | 106 rows
                                                                          Ln 3        Col 1
```

Figure 4.9: 106 synsets has (1 →∞) relation between English synset and Arabic synset

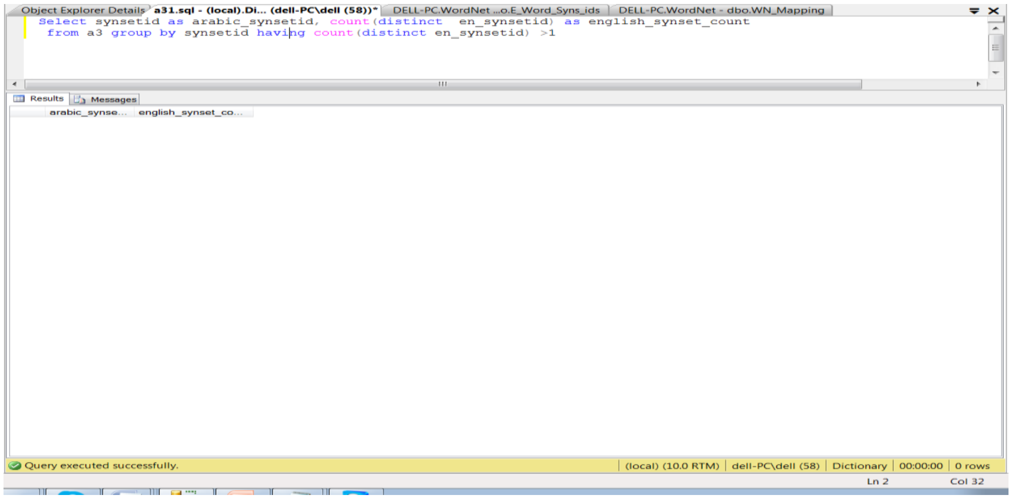
For examples:

- For the English synset {body; consistence; consistency; substance} which has an ID {104941723}, its linked with two different Arabic synsets:
    - {body; consistence; consistency; substance} := {إِتِّسَاق؛ تَمَاسُك؛ تَنَاسُق؛ تَرَابُط}
    - {body; consistence; consistency; substance} := {بَدَن }
- For the English synset {circumstance; condition; consideration }which has an ID {105831106}, its linked with two different Arabic synsets:
    - {circumstance; condition; consideration}:= {شَرْط؛ اِعْتِبَار ؛ مُرَاعَاة}
    - {circumstance; condition; consideration}:= {شَرْط}
- For the English synset {measure; measurement; measuring; mensuration}
    - which has an ID {100998911}, its linked with two different Arabic synsets:
    - {measure; measurement; measuring; mensuration}:={ إِجْرَاء؛ اِعْدَاد؛ تَدْبِير؛ خُطْوَة}
    - {measure; measurement; measuring; mensuration}:={قِيَاس }

0 synsets out of 8,799 have one Arabic synset, which is mapped to many English synsets i.e., the relation is (1 →1) for this part. Figure 4.13 shows the query screenshot which applied mapped synsets to get the result (1 →1).

The query to find the number of Arabic Synset which is mapped to many English synset:

```
Select synsetid as arabic_synsetid, count(distinct en_synsetid) as english_synset_count
        from dbo.a3 group by synsetid having count(distinct en_synsetid) >1
```



Object Explorer Details
a31.sql - (local).Di... (dell-PC\dell (58))*
DELL-PC.WordNet ...o.E_Word_Syns_ids
DELL-PC.WordNet - dbo.WN_Mapping

```
Select synsetid as arabic_synsetid, count(distinct  en_synsetid) as english_synset_count
 from a3 group by synsetid having count(distinct en_synsetid) >1
```

arabic_synse...	english_synset_co...

```
Query executed successfully.    (local) (10.0 RTM) | dell-PC\dell (58) | Dictionary | 00:00:00 | 0 rows
                                                          Ln 2          Col 32
```

Figure 4.10: 0 synsets has (1 →1) relation between Arabic synset and English synset

4.4. Mapping the Arabic synsets to the EWN concepts automatically

In this section, we describe the steps followed to automatically map the Arabic synsets that resulted from the Automatic Arabic thesaurus algorithm SynsetGenerator application to the existing EWN concepts. In what follows, we describe the Mapping algorithm and then we demonstrate our implementation of this algorithm.

4.4.1. Mapping the Arabic synsets to the EWN concepts algorithm

Given an Arabic thesaurus file which contains Arabic synsets and their equivalent English synsets such that $\{a_1, a_2, \ldots, a_n\} := \{e_1, e_2, \ldots, e_k\}$where a_n refers to an Arabic words, e_k refers to an English words, the mapping algorithm will map the Arabic synsets to the existing EWN concepts through the English synset.

The mapping process is done by using cosine similarity measurement, which measures the similarity between two vectors (English synset and EWN synset) by measuring the cosine of the angle between them. If the ratio is high, then similarity is high. The highest similarity ratio is 1, which means they are the same.

Cosine similarity converts the string to a vector to find the similarity ratio between the two synsets. The mapping process done by listing the distinct words then counting the frequency of each word in the list. The counter of each word is incremented by one each time it found in the original string. These counters are then saved in an array. The process will be repeated for the second string. Finally, by formatting the two vectors, we can substitute these vectors in the cosine similarity equation that is represented below [44]:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

To understand how cosine similarity builds the vectors, suppose the following example [48] which has text1 and text 2 as shown below:

Text 1: Jana loves me more than Jeni loves me

Text 2: Jwana likes me more than Jine loves me

The first step is to list the distinct words from both texts (the word order is not important):

me Julie loves Linda than more likes Jane

Then count the number of times each of these words appears in each text to build two vertical vectors of counts as in table 4.5:

Num	Distinct Words	Counts in Text 1	Counts in Text 2
1.	Me	2	2
2.	Jana	1	1
3.	Likes	0	1
4.	Loves	2	1
5.	Jwana	0	1
6.	Jeni	1	0
7.	Than	1	1
8.	More	1	1

Table 4.5: Counter of words

We are not interested in the words themselves though. We are interested only in two vertical vectors of counts. The two vectors are:

A: [2, 1, 0, 2, 0, 1, 1, 1]

B: [2, 1, 1, 1, 1, 0, 1, 1]

By applying the two vectors in the cosine similarity equation to find the similarity, we can find that the angle between the two vectors is 0.822.Thus, we can conclude that the similarity between the two texts is 0.822.

The mapping of the Arabic synsets to the EWN synsets is done by listing all EWN synsets (from the EWN) which have common English words with the English synset (from the Arabic Thesaurus) and then giving each one a weight which computed using the cosine similarity measurement. The highest EWN synset weight is then mapped to the Arabic synset. To be more familiar, suppose we have the following example, which has synsets from Arabic Thesaurus resulted file, Arabic synsets and English synsets:

$\{a_1, a_2, \ldots, a_n\} := \{e_1, e_2, \ldots, e_k\}$

Where a_n refers to an Arabic word and e_k refers to an English word.

The following steps summarize the Mapping algorithm:

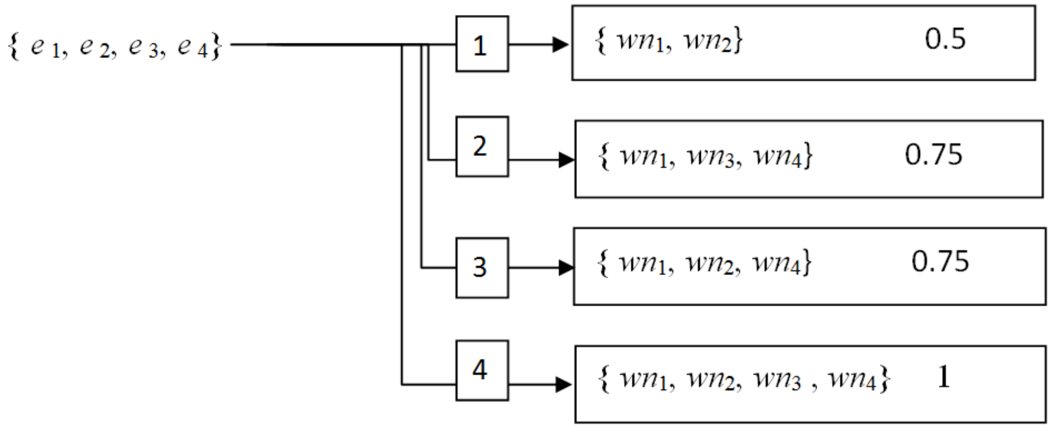
Step-1: For each English synset$\{e_1, e_2, \ldots e_k\}$, find and list of all EWN synsets$\{wn_1, wn_2, \ldots, wn_m\}$which are similar to English synsets using cosine similarity.

Step-2: Map the English synset to the highest EWN synset similarity ratio.

Step-3: Map the Arabic synset, which is equivalent to the English synset in the Arabic thesaurus file, to the mapped EWN synset in **step-2**.

Step-4: Repeat **step-1, step-2** and **step-3** for all English synsets in the Arabic thesaurus file, which resulted from **SynsetGenerator** application.

To further illustrate the Mapping algorithm suppose we have $\{e_1, e_2, e_3, e_4\}$English synset from the Arabic thesaurus file. Cosine similarity lists all EWN synsets which have a similarity to the selected English synset. In this example, four EWN synsets are found, each synset has a cosine similarity value. The highest value is then mapped to the English synset part $\{e_1, e_2, e_3, e_4\}$.


$\{e_1, e_2, e_3, e_4\}$
1
$\{wn_1, wn_2\}$ 0.5
2
$\{wn_1, wn_3, wn_4\}$ 0.75
3
$\{wn_1, wn_2, wn_4\}$ 0.75
4
$\{wn_1, wn_2, wn_3, wn_4\}$ 1

The fourth EWN synset $\{wn_1, wn_2, wn_3, wn_4\}$ has the highest cosine similarity value which is equal to one. This EWN synset will be mapped to the English synset:

$$\{e_1, e_2, e_3, e_4\} := \{wn_1, wn_2, wn_3, wn_4\}$$

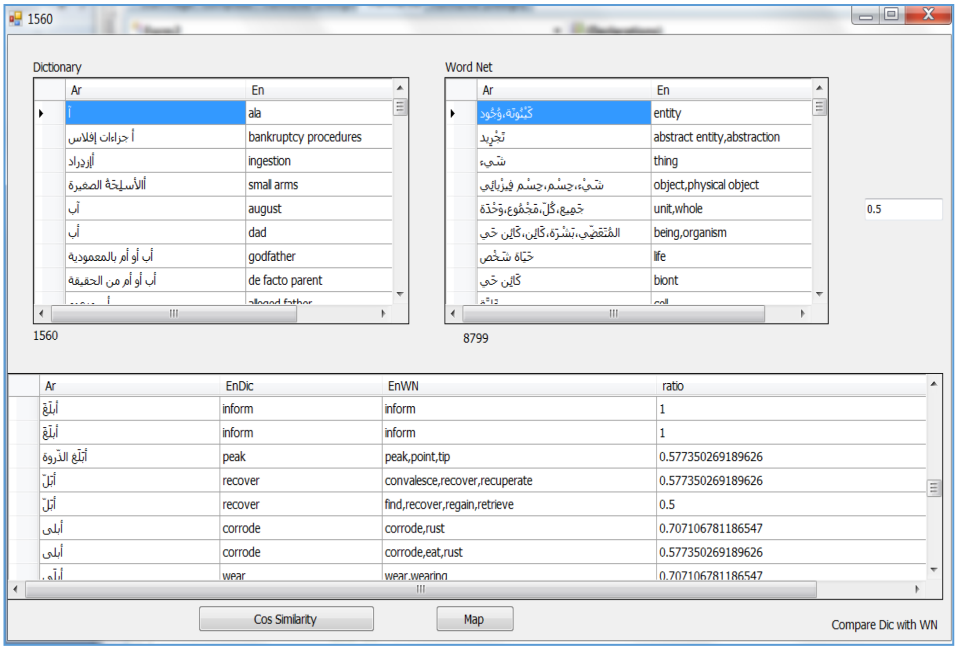
Then the Arabic synset is mapped to the EWN synset:

$$\{a_1, a_2, a_3, a_4, a_5\} := \{e_1, e_2, e_3, e_4\} := \{wn_1, wn_2, wn_3, wn_4\}$$

The process will be repeated for all Arabic Thesaurus file that resulted from **SynsetGenerator** application.

4.4.2. Mapping the Arabic synsets to the EWN concepts implementation

Figure 4.1 shows the graphical interface for the Mapping application. The similarity value should be applied in order to map the Arabic synsets to the highest EWN synsets. This is accomplished by mapping the English synsets to the highest EWN synsets similarity ratio. This ratio will be greater or equal to the selected similarity value. If the EWN synset has a similarity ratio less than the similarity selected value, then it will not be listed in the EWN synsets. If we have two EWN synsets, which have the same similarity ratio with the English synset, then any of them will be mapped to Arabic synset. After that, the mapping synsets are exported to a text file.



Dictionary

Ar	En
آ	ala
أ جزاءات إفلاس	bankruptcy procedures
أإزدِراد	ingestion
أالأسلِحَةُ الصغيرة	small arms
آب	august
أب	dad
أب أو أم بالمعمودية	godfather
أب أو أم من الحقيقة	de facto parent
	alleged father

1560

Word Net

Ar	En
كَيْنُونَة،وُجُود	entity
تَجْرِيد	abstract entity,abstraction
شَيْء	thing
شَيْء،جِسْم،جِسْم فِيزْيائِي	object,physical object
جَمِيع،كُلّ،مَجْمُوع،وَحْدَة	unit,whole
المُتَقَضِّي،بَشَرَة،كَائِن،كَائِن حَي	being,organism
حَيَاة شَخْص	life
كَائِن حَي	biont
	cell

8799

0.5

Ar	EnDic	EnWN	ratio
أبلَغَ	inform	inform	1
أبلَغَ	inform	inform	1
أبلَغَ الذّروة	peak	peak,point,tip	0.577350269189626
أبَلّ	recover	convalesce,recover,recuperate	0.577350269189626
أبَلّ	recover	find,recover,regain,retrieve	0.5
أبلى	corrode	corrode,rust	0.707106781186547
أبلى	corrode	corrode,eat,rust	0.577350269189626
	wear	wear,wearing	0.707106781186547

Figure 4.12: Mapping application interface

Through this interface, we put the similarity value equal to, e.g., 0.5, only the EWN with similarity value greater than or equal to 0.5 will be listed. The EWN with a value less than 0.5 will be ignored.

Once we press the cosine similarity button, the similarity computation (depends on cosine similarity measurement) between each English synsets from the left block (these synsets come from the execution of the **SynsetGenerator** application) and all EWN synsets in the right block (these synsets are from the linked WN application) will be started. Cosine similarity builds two count vectors for the two synsets, these vectors are constructed by the string matching between the two synsets, i.e., it measures the similarity between two vectors by measuring the cosine angle between them [44].

In the third block, each dictionary synset with all EWN synsets that have a similarity value more than the similarity value which we entered will be listed. The map button will map Arabic synset to the highest EWN synset similarity value, then export the result $\{a_1,a_2, a_3, a_4, a_5\} := \{ wn_1, wn_2, wn_3, wn_4\}$ synsets to a text file.

The percentage of mapped synsets is affected by two factors:

- The selection of similarity value, the maximum value is 1, which means exact match, and the minimum value is equal to 0, which means no similarity.
- The type of bilingual dictionary, which we want to map. For example, if we have Musical bilingual dictionary, then, for sure, the percentage of mapped synsets will be small.

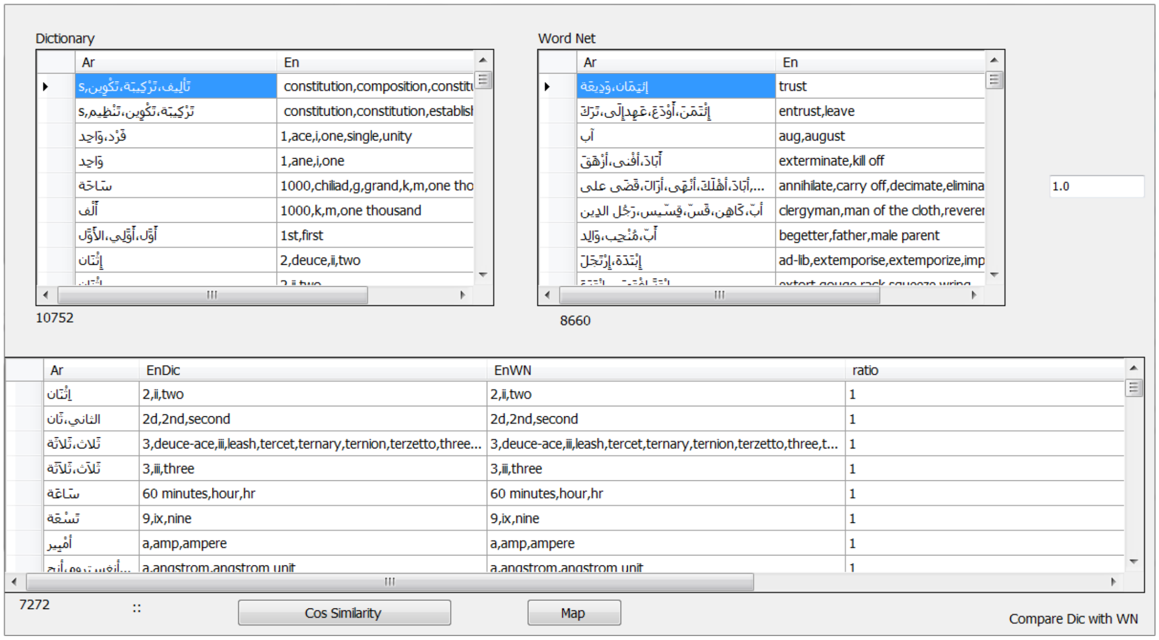
4.5. Evaluating SynsetGenerator algorithm and WN synsets using mapping algorithm

As we presented in the previous section, the Mapping application maps the Arabic synsets to their equivalent EWN concepts through the similarity computation between the English synsets and the EWN synsets.

This section presents how we can use the mapping algorithm to find the number and percentage of exact WN synset that are generated by **the SynsetGenerator algorithm**(As we presented in the previous section, the Mapping application maps the Arabic synsets to their equivalent EWN concepts through the similarity computation between the English synsets and the EWN synsets.

This section presents how we can use the mapping algorithm to find the number and percentage of exact synset that are generated by **the SynsetGenerator algorithm**(10752 synsets) compared to the original linked WN synsets (8660 synsets).

The similarity computation in the mapping algorithm compares WN synsets, which are generated by WN synset application and WN synsets, which are generated by **SynsetGenerator** application. Putting the similarity value equal to 1 will list only the exact matched synsets, which means only the WN synsets that are generated by **SynsetGenerator** implantation and exactly the same as WN synsets that are generated by the WN application will be listed as seen in figure 4.15 which shows the exact WN synset match.



Dictionary

	Ar	En
▸	تَأْلِيف،تَرْكِيبَة،تَكْوِين,s	constitution,composition,constitu
	تَرْكِيبَة،تَكْوِين،تَنْظِيم,s	constitution,constitution,establisl
	فَرْد،وَاحِد	1,ace,i,one,single,unity
	وَاحِد	1,ane,i,one
	سَاحَة	1000,chiliad,g,grand,k,m,one tho
	أَلْف	1000,k,m,one thousand
	أَوَّل،أَوَّلِي،الأَوَّل	1st,first
	إِثْنَان	2,deuce,ii,two

10752

Word Net

	Ar	En
▸	إِئْتِمَان،وَدِيعَة	trust
	إِئْتَمَنَ،أَوْدَعَ،عَهِدإِلَى،تَرَكَ	entrust,leave
	آب	aug,august
	أَبَادَ،أَفْنَى،أَزْهَقَ	exterminate,kill off
	أَبَادَ،أَهْلَكَ،أَنْهَى،أَزَالَ،قَضَى على,...	annihilate,carry off,decimate,elimina
	أَبّ،كَاهِن،قَسّ،قِسّيس،رَجُل الدِين	clergyman,man of the cloth,reverer
	أَبّ،مُنْجِب،وَالِد	begetter,father,male parent
	إِبْتَدَهَ،إِرْتَجَلَ	ad-lib,extemporise,extemporize,imp

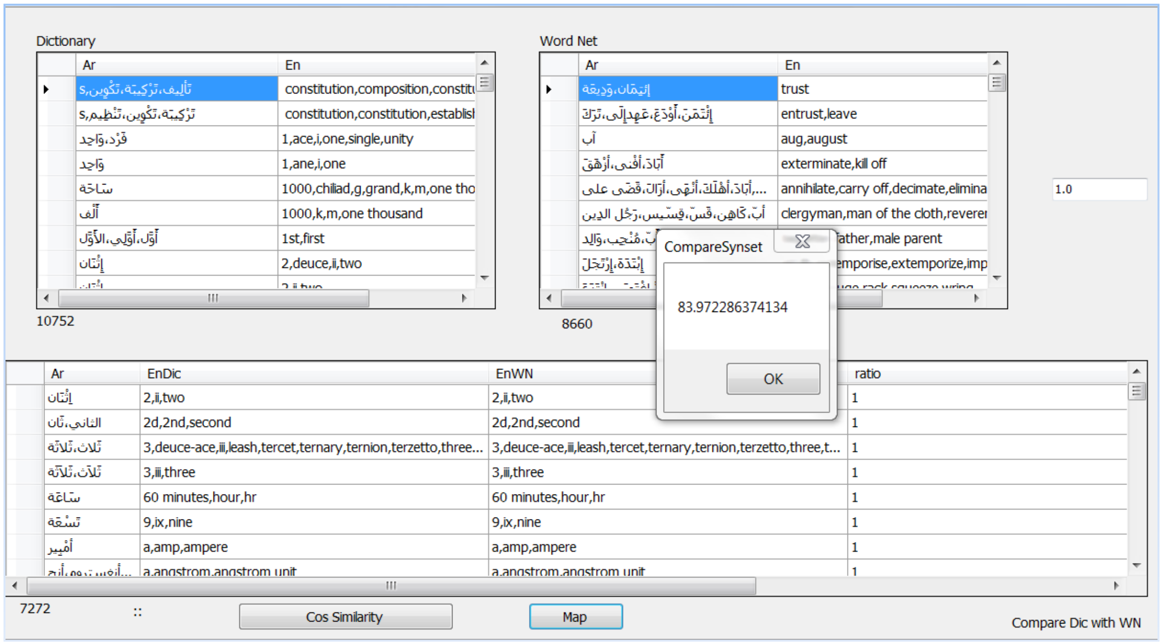
8660

1.0

	Ar	EnDic	EnWN	ratio
	إِثْنَان	2,ii,two	2,ii,two	1
	الثاني،ثَان	2d,2nd,second	2d,2nd,second	1
	ثَلاث،ثَلاثَة	3,deuce-ace,iii,leash,tercet,ternary,ternion,terzetto,three...	3,deuce-ace,iii,leash,tercet,ternary,ternion,terzetto,three,t...	1
	ثَلاث،ثَلاثَة	3,iii,three	3,iii,three	1
	سَاعَة	60 minutes,hour,hr	60 minutes,hour,hr	1
	تِسْعَة	9,ix,nine	9,ix,nine	1
	أَمْبِير	a,amp,ampere	a,amp,ampere	1

7272 ::

Figure 4.13: Finding the exact WN synset match using mapped algorithm



CompareSynset

83.972286374134

Dictionary

	Ar	En
▸	تَأْلِيف،تَرْكِيبَة،تَكْوِين,s	constitution,composition,constitu
	تَرْكِيبَة،تَكْوِين،تَنْظِيم,s	constitution,constitution,establisl
	فَرْد،وَاحِد	1,ace,i,one,single,unity
	وَاحِد	1,ane,i,one
	سَاحَة	1000,chiliad,g,grand,k,m,one tho
	أَلْف	1000,k,m,one thousand
	أَوَّل،أَوَّلِي،الأَوَّل	1st,first
	إِثْنَان	2,deuce,ii,two

10752

Word Net

	Ar	En
▸	إِئْتِمَان،وَدِيعَة	trust
	إِئْتَمَنَ،أَوْدَعَ،عَهِدإِلَى،تَرَكَ	entrust,leave
	آب	aug,august
	أَبَادَ،أَفْنَى،أَزْهَقَ	exterminate,kill off
	أَبَادَ،أَهْلَكَ،أَنْهَى،أَزَالَ،قَضَى على,...	annihilate,carry off,decimate,elimina
	أَبّ،كَاهِن،قَسّ،قِسّيس،رَجُل الدِين	clergyman,man of the cloth,reverer

8660

1.0

	Ar	EnDic	EnWN	ratio
	إِثْنَان	2,ii,two	2,ii,two	1
	الثاني،ثَان	2d,2nd,second	2d,2nd,second	1
	ثَلاث،ثَلاثَة	3,deuce-ace,iii,leash,tercet,ternary,ternion,terzetto,three...	3,deuce-ace,iii,leash,tercet,ternary,ternion,terzetto,three,t...	1
	ثَلاث،ثَلاثَة	3,iii,three	3,iii,three	1
	سَاعَة	60 minutes,hour,hr	60 minutes,hour,hr	1
	تِسْعَة	9,ix,nine	9,ix,nine	1
	أَمْبِير	a,amp,ampere	a,amp,ampere	1

7272 ::

Figure 4.14: The percentage of exact WN synset matching using mapping algorithm

Table 4.6 shows the number of WN synsets used in the experiment (8660 synsets), and the number of synsets that our algorithm generated 10752 synsets. From these synsets the correctly generated synsets are 7272 synsets. This means that the correctly mapped synsets ratios is 7272/8660 = 84 %.

Number of WN synsets used in the experiment	8660 synsets
Number of synsets generated by the **SynsetGenerator**	10752synsets
The number of correctly generated synsets	7272 synsets
Mapped synsets ratio	7272/8660 = 84 %

Table 4.6:The number of the exacted match WN synsets ,and their mapped synsets ratio

Table 4.7 shows the percentages for mapped WN synsets for different values of cosine similarity (1.0, 0.9, 0.8, 0.7, 0.6). We can infer from the table that decreasing the value of cosine similarity will increase the percentages mentioned earlier i.e., decreasing similarity will give tolerance to the application and also the returned synsets.

Cosine Similarity	Number of Generated Correct Synsets	Mapped synsets ratio
1.0	7272 synsets	7272/8660 = **84%**
0.9	7644 synsets	7644/8660 = **88%**
0.8	8015 synsets	8015/8660= **93%**
0.7	8197 synsets	8197/8860= **95%**
0.6	8689 synsets	8689/8660= **100.3%**

Table 4.7: Mapped synsets ratios with different cosine similarities

The major issue affecting the mapped synsets ratio in the above table is that the **SynsetGenerator** generated more results than expected (10,752 – 7,272 = 3,480). When the cosine similarity is equal to 0.6 the mapped synsets ratio was 100.3% which means that we mapped all WN synsets in addition to "extra" synsets (3,480 synsets).These extra synsets can be incorrect synsets that we can ignore and drop or could be right and our algorithm missed them, or this can be caused by the quality of AWN itself as explained earlier in this thesis. In what follows, we provide two evaluations to further investigate these extra generated synsets. Table 4.8 provides statistics regarding the 3,480 extra generated synsets, and whether their length is one word, two, three, four, or more words. In Table 4.9, we show results after repeating the same procedure we did in table 4.7, but now we removed each synset that is a subset of another synset and removed synsets with length equal to one like $\{a_1\}:=\{wn_1\}$, $\{a_1\}:=\{e_1\}$ from both WN synsets and

SynsetGenerator synsets. We removed 2,993 synsets from WN synsets and about 3,871 synsets from **SynsetGenerator** synsets. The remainder 5,667 synsets (8,660 – 2,993=5,667) and 6,881 synsets (10,768 – 3,871 = 6,881) are evaluated again.

Num of words in the extra generated synsets	
One word / English synset	7
two words / English synset	1177
three words / English synset	1814
Four words / English synset	77
More than four word /English synset	405

Table 4.8: Statistics about the extra-generated synsets

Cosine Similarity	Number of Generated Correct Synsets	Mapped synsets ratio
1.0	5173 synsets	5173/5667= **91.2%**
0.9	5175 synsets	5175/5667= **91.3%**
0.8	5366 synsets	5366/5667= **95%**
0.7	5415 synsets	5415/5667= **96%**
0.6	5747 synsets	5747/5667= **101%**

Table 4.9: Mapped synsets ratios with different cosine similarities ratios after removing subsets and synsets with one word length

Discussion:

In this part we discuss and explain the results that we obtained in table 4.7.The **SynsetGenerator** algorithm generates 10,752 synsets which exceeds the number of original WN synsets (8,660 synsets) in about3,480 synsets. The exact matched synsets number is 7,272 synsets. The reasons that affect our results and caused the algorithm not to regenerate the other synsets (1,388 synsets) are: **Firstly**, our **SynsetGenerator** synsets builds the synsets in a systematic way while WN synsets are built manually, hence we are comparing synsets that are produced by different approaches. **Secondly,** the WN synsets have many synsets that are subsets from each other such as { أَدْرَكَ،رَأَى }:= { catch, find out } which is a subset from { أَدْرَكَ،أمْسَكَ،رَأَى }:={ catch, find out, get}. **Thirdly**, there are synsets in WN has only one word in Arabic and English such as { نَشِط }:={active} which may require a special treatment. **Finally**, the cosine similarity approach may drop some exact WN synsets as a result of the cosine similarity calculations. The cosine similarity approach compares the two strings in terms of word by word, while programmatically we compare the two strings character by character, so if we have a synset that has a cosine similarity equal to 0.99999 it will not be listed as an exact WN synset.

In addition, the cosine similarity ratio depends on the number of words inside the two compared synsets i.e., if we have a synset that has four words such as {carry on, continue, go on, proceed} and one word is missed from the compared synset such as {carry on, continue, go on} the cosine similarity ratio will be 0.866, while if we have a synset with seven words such as {cloud, befog, becloud, obscure, obnubilate, haze over, mist} and one word is missed from the compared synset such as {befog, becloud, obscure, obnubilate, haze over, mist } the cosine similarity ratio will be different from the synset with four words and it will be 0.926.

In order to resolve these problems we remove the subsets and the synsets with one word length $\{a_1\}$:= $\{ wn_1\}$from both **SynsetGenerator** synsets and the original WN synsets, then evaluate again, These two steps improve the mapping algorithm as the **Mapped synsets ratio** becomes **91.2%** instead of **84%** as seen in table 4.9.

Chapter 5

5. Conclusions and Future work

In this chapter we summarize our idea, work, and results obtained. Moreover, we discuss future work that can be added to the algorithm to further improve results and expand the idea.

5.1. Conclusions

Our contributions in this thesis can be summarized as follows: **firstly**, we implemented the SynsetGenerator algorithm which builds an Arabic thesaurus file automatically as $\{a_1, a_2, ... a_n\} := \{e_1, e_2, ... e_k\}$ from the Arabic-English bilingual dictionary, where a_1,a_2, …, a_n, are the Arabic synonyms that has the same meaning ande_1, e_2, …, e_k are the English synonyms that has the same meaning and equivalence to the Arabic synonyms. **Secondly**, we evaluated the SynsetGenerator algorithm through conducting an experiment that aim at building the WN synsets using the algorithm. We used the cosine similarity approach to compare the generated synsets with the WN. The results were promising as the algorithm built about 84% of the WN synsets. In order to find the reasons that caused our algorithm to not create the other synsets, we tested the WN three assumptions which we supposed that they are valid for the WN, we found that the WN has some problems since the three assumptions are not fulfilled by WN. For example, each set of synonyms grouped into a synset in both AWN and EWN may have different IDs, there are subsets in the AWN and EWN that have relation which is not one-to-one between the linked AWN and EWN synsets that explains the reason why we did not regenerated all the WN synsets. The Final step in the thesis is to map the Arabic thesaurus file to the English WN. That is, the result will be a set of Arabic synsets mapped into WN synsets as $\{a_1, a_2, …, a_n\} := \{wn_1, wn_2, …, wn_m\}$, we use the cosine similarity approach for this purpose.

5.2. Future work

The current results we obtained from the conducted experiment with different values of cosine similarity provides a promising value of the algorithm to be used as a synonyms generator. To further improve the performance and accuracy of the algorithm proposed in this thesis we suggest the use of artificial intelligent methods and techniques such as machine learning tools to make the algorithm smarter and enable it to scale and adapt to different situations.

Moreover, we can test the algorithm to see if it can be applied to other languages other than Arabic and English. An evaluation of the algorithm using other languages is an interesting topic to pursue and to investigate. In addition, the algorithm can be exploited for the purpose of evaluating Arabic-English dictionaries by comparing it with respect to WordNet using the Mapping algorithm mentioned in section 4.4. Furthermore, building a website that contains the different functionalities provided by the developed application in this thesis such as the synonyms generation and synsets mapping would be of a great assist to other researchers who desire to exploit these functionalities and also it would useful to interested individuals from different fields. Try to build again the Mapping algorithm using Jaccard similarity approach, then compare the performance and the mapping percentage which we got from both different approaches .

References

1. Farghaly, A. and K. Shaalan, Arabic natural language processing: Challenges and solutions. ACM Transactions on Asian Language Information Processing (TALIP), 2009. 8(4): p. 14.
2. Boukhatem, N., The Arabic Natural Language Processing: Introduction and Challenges. International Journal of English Language & Translation Studies, 2014. 2(3): p. 106-112.
3. Chowdhury, G.G., Natural language processing. Annual review of information science and technology, 2003. 37(1): p. 51-89.
4. Crouch, C.J. and B. Yang. Experiments in automatic statistical thesaurus construction. in Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval. 1992. ACM.
5. Hawker, S. and C. Soanes, Compact Oxford English Dictionary of Current English. 2005: Oxford University Press.
6. Al-Qabbany, A., A. Al-Salman, and A. Almuhareb. An automatic construction of Arabic similarity thesaurus. in Proceedings of the 3rd IEEE International Conference on Arabic Language Processing (CITALA2009). 2009.
7. Jarrar, M., Lecture notes on Arabic Ontology. Artificial Intelligence 2014.
8. قاموس المعاني. [cited 2014 Jun 03]; Available from: http://www.almaany.com/thesaurus.php.
9. Collins English Thesaurus. [cited 2014 Jun 03]; Available from: http://collins-english-thesaurus.en.softonic.com/.
10. Magnini, B., et al. Multilingual lexical knowledge bases: Applied WordNet prospects. in Proceedings of the International Workshop on ‘‘The Future of the Dictionary. 1994. Citeseer.
11. Miller, G.A., et al., Introduction to wordnet: An on-line lexical database*. International journal of lexicography, 1990. 3(4): p. 235-244.
12. Derwojedowa, M., et al. Words, concepts and relations in the construction of Polish WordNet. in Proceedings of the Global WordNet Conference, Seged, Hungary. 2008.
13. WorldWeb. WorldWeb: Free English dictionary and Thesaurus. [cited 2014 June 03]; Available from: http://wordweb.info/free/.
14. Jarrar, M., WordNet EuroWordNet, and Global WordNet. 2014.
15. Miller, G.A., WordNet: a lexical database for English. Communications of the ACM, 1995. 38(11): p. 39-41.
16. Pedersen, T., S. Patwardhan, and J. Michelizzi. WordNet:= Similarity: measuring the relatedness of concepts. in Demonstration papers at HLT-NAACL 2004. 2004. Association for Computational Linguistics.
17. Elkateb, S., et al. Arabic WordNet and the challenges of Arabic. in Proceedings of Arabic NLP/MT Conference, London, UK. 2006. Citeseer.
18. Weisscher, A. AWN Browser. 2014; Available from: http://globalwordnet.org/arabic-wordnet/awn-browser/.
19. Jarrar, M. Position paper: towards the notion of gloss, and the adoption of linguistic resources in formal ontology engineering. in Proceedings of the 15th international conference on World Wide Web. 2006. ACM.
20. Gruber, T.R., Toward principles for the design of ontologies used for knowledge sharing? International journal of human-computer studies, 1995. 43(5): p. 907-928.

21. Jarrar, M., Towards Effectiveness and Transparency in e-Business Transactions, in Semantic Web Methodologies for E-Business Applications. 2008, IGI Global. p. 127-149.
22. Jarrar, M. and R. Meersman, Scalability and knowledge reusability in ontology modeling. STAR, 2002: p. 09.
23. Jarrar, M. Building a Formal Arabic Ontology (Invited Paper). in proceedings of the Experts Meeting on Arabic Ontologies and Semantic Networks. Alecso, Arab League. Tunis. 2011.
24. Jarrar, M., A. Deik, and B. Farraj, Ontology-based Data and Process Governance Framework-The Case of e-Government Interoperability in Palestine. 2011.
25. Jarmasz, M. and S. Szpakowicz, Roget's thesaurus: A lexical resource to treasure. arXiv preprint arXiv:1204.0258, 2012.
26. Michelbacher, L., et al. Building a Cross-lingual Relatedness Thesaurus using a Graph Similarity Measure. in LREC. 2010.
27. Yang, D. and D.M. Powers. Automatic thesaurus construction. in Proceedings of the thirty-first Australasian conference on Computer science-Volume 74. 2008. Australian Computer Society, Inc.
28. Hirschman, L., R. Grishman, and N. Sager, Grammatically-based automatic word class formation. Information Processing & Management, 1975. 11(1): p. 39-57.
29. Black, W., et al. Introducing the Arabic wordnet project. in Proceedings of the 3rd International WordNet Conference (GWC-06). 2006.
30. Shamsfard, M., et al. Semi automatic development of farsnet; the persian wordnet. in Proceedings of 5th Global WordNet Conference, Mumbai, India. 2010.
31. Atserias, J., et al., Combining multiple methods for the automatic construction of multilingual WordNets. arXiv preprint cmp-lg/9709003, 1997.
32. Vetulani, Z., M. Kubis, and T. Obrêbski. PolNet-Polish WordNet: Data and Tools. in LREC. 2010.
33. Pala, K. and P. Smrž, Building czech wordnet. Romanian Journal of Information Science and Technology, 2004. 7(1-2): p. 79-88.
34. Pustejovsky, James, and Amber Stubbs. Natural language annotation for machine learning. " O'Reilly Media, Inc.", 2012.
35. Buckland, M.K. and F.C. Gey, The relationship between recall and precision. JASIS, 1994. 45(1): p. 12-19.
36. Powers, D.M., Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.
37. Ye, N., et al. Optimizing F-measures: a tale of two approaches. in Proceedings of the International Conference on Machine Learning. 2012.
38. KantanMT. What is F-Measure? [cited 2014 Jun 24]; Available from: http://www.kantanmt.com/whatisfmeasure.php.
39. Dagan, I., L. Lee, and F.C. Pereira, Similarity-based models of word cooccurrence probabilities. Machine Learning, 1999. 34(1-3): p. 43-69.
40. An Introduction to Mathematical Taxonomy. B. S. Everitt and G. Dunn. Dover Publications. ISBN-13: 978-0486435879. 1982
41. Keshavarzi, M., M. Dehghan, and M. Mashinchi, Applications of classification based on similarities and dissimilarities. Fuzzy Information and Engineering, 2012. 4(1): p. 75-91.
42. Dehak, N., et al. Cosine Similarity Scoring without Score Normalization Techniques. in Odyssey. 2010.
43. Qian, G., et al. Similarity between Euclidean and cosine angle distance for nearest neighbor queries. in Proceedings of the 2004 ACM symposium on Applied computing. 2004. ACM.

44. Juricic, V. Evaluation of similarity metrics for programming code plagiarism detection method. in CECIIS-2011. 2011.

45. Why distance is a bad idea. [cited 2015 3 March]; Available from: http://slidewiki.org/slide/7598.

46 Kim, Myoung-Cheol, and Key-Sun Choi. "A comparison of collocation-based similarity measures in query expansion." Information processing & management 35.1 (1999): 19-30.

47. Niwattanakul, Suphakit, et al. "Using of Jaccard coefficient for keywords similarity." Proceedings of the International MultiConference of Engineers and Computer Scientists. Vol. 1. 2013.

48. Cohen, William, Pradeep Ravikumar, and Stephen Fienberg. "A comparison of string metrics for matching names and records." Kdd workshop on data cleaning and object consolidation. Vol. 3. 2003.