Towards the Enrichment of Arabic WordNet with Big Corpora

Georges Lebboss[1], Gilles Bernard[1], Noureddine Aliane[1] and Mohammad Hajjar[2]
[1]*LIASD, Paris 8 University, Paris, France*
[2]*Lebanese University, IUT, Saida, Lebanon*

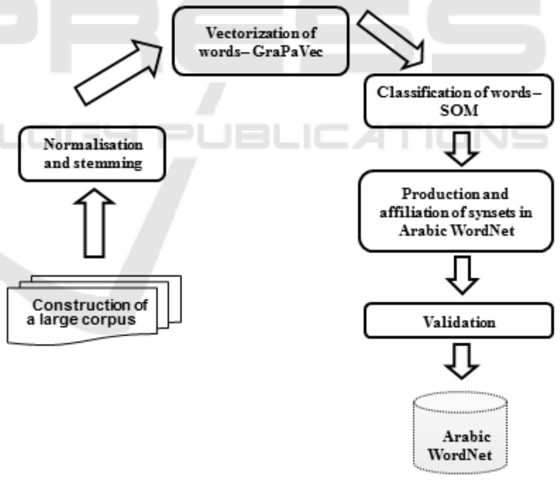
Keywords: Semantic Relations, Semantic Arabic Resources, Arabic WordNet, Synsets, Arabic Corpus, Data Preprocessing, Word Vectors, Word Classification, Self Organizing Maps.

Abstract: This paper presents a method aiming to enrich Arabic WordNet with semantic clusters extracted from a large general corpus. The Arabic language being poor in open digital linguistic resources, we built such a corpus (more than 7.5 billion words) with ad-hoc tools. We then applied GraPaVec, a new method for word vectorization using automatically generated frequency patterns, as well as state-of-the-art Word2Vec and Glove methods. Word vectors were fed to a Self Organizing Map neural network model; the clusterings produced were then compared for evaluation with Arabic WordNet existing synsets (sets of synonymous words). The evaluation yields a F-score of 82.1% for GrapaVec, 55.1% for Word2Vec's Skipgram, 52.2% for CBOW and 56.6% for Glove, which at least shows the interest of the context that GraPaVec takes into account. We end up by discussing parameters and possible biases.

1 INTRODUCTION

The Arabic language is poor in open digital linguistic resources, especially semantic ones. Work in the field of automatic semantic analysis is not as developed as for European languages; improving such resources is an important goal for researches on Arabic language and semantics. Among these resources we choose Arabic WordNet (Black et al., 2006; Rodriguez et al., 2008; Regragui et al., 2016), an open semantic database where lexical items are organized in synsets (sets of synonymous words), linked by semantic relationships, based on WordNet (Miller, 1995), now version 2.1 (Miller et al., 2005). Arabic WordNet (hereafter AWN) is still poor in words and synsets and needs to be enriched.

The end-to-end system presented here (figure 1) generates semantic word clusters computed from a large general corpus (Lebboss, 2016). Existing methods (subsection 2.1) are based on dictionaries (either digitized paper ones or database dictionaries as Wiktionary), on translation and aligned multilingual corpus, on WordNets and ontologies, on morphological parsing or on combinations of those resources. There are not any methods based on large general corpora.


Construction of a large corpus
Normalisation and stemming
Vectorization of words– GraPaVec
Classification of words– SOM
Production and affiliation of synsets in Arabic WordNet
Validation
Arabic WordNet
Figure 1: Global view of our system.

Available Arabic corpora are small, and researchers working on Arabic corpora usually have had to devise their own. Our first step was to build the biggest possible open corpus (section 3), keeping in mind that it should be dynamically computed (so as to expand as much as possible as resources grow) and that the building tool should be freely available for researchers. The corpus built contains more than 7.5 billion words; it is by large the biggest one ever made for Arabic language.

Arabic corpora usually are preprocessed by orthographic normalization and lemmatization. Arabic lemmatization has been thoroughly analyzed by Al Hajjar in his PhD thesis (Al Hajjar, 2010); we choose the lemmatizer that according to his evaluation yielded the best results (Khoja et al., 2001).

The main issue is word vectorization. Methods considered here are based upon distributional properties of words (subsection 2.2). The main characteristic of GraPaVec as opposed to the state-of-the-art methods is that the context taken into account is the surrounding pattern of high frequency words rather than a window of neighbouring lexical items or skip-gram of lexical items (section 4). In other words, we keep what others throw away and throw away what others keep.

Vectors are fed to a clustering algorithm. We have chosen here the neural network model Self Organizing Maps (Kohonen, 1995), because of two advantages: minimization of misclassification errors (misclassified items go to adjacent clusters) and easy visualization of the results. Those results are then evaluated by comparison with AWN synsets (section 6).

2 RELATED WORK

2.1 Building Semantic Resource

In 2008, three methods were proposed by the AWN team.

One (Rodriguez et al., 2008) builds a bilingual lexicon of <English word, Arabic word, POS> tuples from several publicly available translation resources. It merges in one set the base concepts of EuroWordNet (1024 synsets) and Balkanet (8516 synsets). Keeping only the tuples whose English word was included in the merged set, they produced <English word, Arabic word, Concept> tuples. Arabic words linked to the same concept were candidates to enter an synset in AWN. Their candidatures were to be validated by lexicographers; however, as of today only 64.5% have been processed, of which 74.2% were rejected as incorrect.

They obtained better results with another method (Rodriguez et al., 2008), where they generated new Arabic forms by morphological derivation from the words in AWN synsets, controled their existence with databases such as GigaWord non free Arabic corpus, the Logos multilingual translation portal, or New Mexico State University Arabic-English lexicon, and used their translation to link them to WordNet synsets and then back to AWN, to be validated by lexicographers. A similar method was proposed later (Al-Barhamtoshy and Al-Jideebi, 2009); words were morphologically hand-parsed by linguists, then translated and associated to synsets with equivalence relations between the synsets made explicit in the Inter-Lingual Index deep structure (Vossen, 2004).

The third method (Alkhalifa and Rodriguez, 2008) extracted named entities from Arabic Wikipedia, linked them to named entities from the corresponding English Wikipedia page, linked those to named entities from WordNet, and then back to synsets of AWN. Though the result was much better (membership was correct up to 93.3%), the coverage was scarce.

A different approach (Abouenour et al., 2008) exported the entire set of data embedded in AWN into a database integrated with Amine AWN ontology, tapped by a Java module based on Amine Platform APIs. This module used the mapping between English synsets in WordNet and Suggested Upper Merged Ontology (Niles and Pease, 2003) concepts to build the Amine AWN type hierarchy. Then, it added Arabic synonyms based on the links between WordNet synsets and AWN synsets.

Later the same team (Abouenour et al., 2010; Abouenour et al., 2013) used YAGO (Yet Another Great Ontology) from Max-Planck Institute, translating its named entities into Arabic with Google translation, then added them to AWN according to two types of mappings (direct mapping through WordNet, mapping through YAGO relations to AWN synsets).

Abdul Hay's PhD thesis (Abdulhay, 2012) extracted semantic categories from a multilingual aligned corpus with English and two langages from EuroWordNet. If all but Arabic words were members of synsets linked by Inter-Lingual Index, then the Arabic word should also be in a linked synset in AWN. Results were correct up to 84%.

Another team worked on iSPEDAL, a database monolingual dictionary digitizing monolingual paper dictionaries (Al Hajjar, 2010). Two methods have been proposed (Hajjar et al., 2013) for enriching iSPEDAL. One used semi-structured information from plain dictionaries to deduce links (synonymy, antonymy). The other used translation by available resources to and from a foreign language to compute synonymy of Arabic words by correlating their translations. A somewhat similar approach (Abdelali and Tlili-Guiassa, 2013) extracts synonymy and antonymy relationships from Arabic Wiktionary.

Arabase platform (Raafat et al., 2013) aims to integrate every available Arabic semantic resource, from *King Abdulaziz City for Science and Technology* database, to Arabic StopWords Sourceforge resource and AWN. It has, according to the authors, "a good potential to interface with WordNet". Arabase computes by hand-made rules semantic properties of vocalized words[1] and forms a sort of virtual WordNet.

[1]Short vowels are not written in Arabic words in normal

As one can see, researches on Arabic semantic categories has extensively used foreign resources; very little has been done on extracting semantic information from Arabic data alone, and nothing based on an Arabic general corpus.

2.2 Word Vectorization

Structural linguistics (Harris, 1954) postulated that words with similar distributions have similar categories and meanings. Since (Salton et al., 1975) first proposed it, projecting words in vector space has been the first step in many models of word clustering, where semantic properties could be linked to similarities in the distribution of the word vectors.

In such models a distribution is defined by a vector of the contexts of a word. A context is defined by two elements: units and distance to the word. Units can be words, phrases or ngrams, more recently skipgrams (ngrams with "holes"). The distance can be a step function, as in the bag of word model (only words in the same document are nearby contexts), or a function of the number of units separating word and context.

On the resulting matrix $\{\text{words} \times \text{contexts}\}$ (where each component is the frequency of a word in a context), mathematical models have been applied in order to reduce it to a bunch of clusters, from the most simple, tf-idf, to much more complex ones, such as latent semantic analysis, latent Dirichlet allocation, neural networks (various models), linear or bilinear regression models... Clustering models display quite a big variety of reduction methods, which contrasts with the poverty of context variety. Usually units are lexical items or ngrams of lexical items and sets are either documents or fixed-length windows.

Some examples: in Hyperspace Analogue to Language (Lund et al., 1995), context is a fixed-length window of lexical items around the word. In WebSOM (Honkela et al., 1997), a model with two layers of Self Organizing Map neural network, context is a fixed-length window of lexical items around the word. Among the rare exceptions to the lexical items context was a SOM classifier applied to a context of a fixed-length window of grammatical categories to the left of the word (Bernard, 1997).

Word2vec (Mikolov et al., 2013) is a set of two unsupervised neural models, with log-linear classifiers, where words and context vectors are trained simultaneously; in CBOW (Continuous Bag Of Words) context is defined by a fixed-length window of lexical items around the word, in Skipgram lexical items are grouped in skipgrams, that is, ngrams with holes.

use and in a majority of documents.

Glove (Global Vectors for Word Representation) (Pennington et al., 2014) is a global log-bilinear regression model designed by the NLP team at Stanford University. Paraphrasing the authors, this model combines the advantages of the global matrix factorization with windowing local context methods. Context is a fixed-length window of lexical items centered on the word.

Both models represent the state of the art. One should note that though they use fixed-length windows, they use a continuous function for distance, thus introducing a un-bag-of-word approach (even in CBOW) which had rarely been used before.

3 CORPUS

To extract semantic categories from general corpora, a large corpus is needed. We merged the entire available Arabic corpora and found it to be small, even adding Arabic Wikipedia and Wiktionary. This static corpus was the starting point of our large corpus, but the bulk comes from the Alshamela library online resource (http://shamela.ws) and, over all, crawling/converting web sites (more than 120, mostly news web sites) and their documents.

Open-source web crawlers as HTTtrack failed to fit our purpose: no queuing, hard resuming of download, thousands of blank pages in the result and no easy way to convert documents on the fly. We created our own Arabic Corpus Builder, that crawls queues of sites, merges them in plain text format with the outcome of previous corpus, and imports it in a database. It also converts on the fly usual encodings of Arabic characters (Microsoft and MacOS) in unicode. It can be found on https://sites.google.com/site/georgeslebboss.

Our corpus is mostly dynamic as the results of crawling varies in time. In its present state, it contains about 85,000 documents and is described in the following table:

Table 1: General corpus.

Source	Word number	Unique words
Static corpus	207 878 809	3 589 374
Arabic Wikipedia + Arwiktionary	6 242 131	2 376 805
Alshamela library	1 862 000 347	4 007 846
Corpus Builder	5 543 097 123	5 987 391
Total	7 619 218 410	6 894 986

Arabic writing conventions, especially concerning vowels, entail that every word can have several writings (not counting errors). Orthographic normalization is usual in Arabic language processing systems, even though it introduces ambiguities.

As Arabic language has a rather complex morphology, especially in derivation, and as writing conventions do not separate morphemes inside accented words, lemmatization of Arabic text, though difficult, is much more useful than it is in European languages (with the exception of German).

In our case, orthographic normalization and lemmatization are left to the choice of the experimentor. We used the best lemmatizer according to the results of Al Hajjar (Al Hajjar, 2010), Khoja lemmatizer (see section 1).

4 GRAPAVEC

For our own method of word vectorization, we explored the idea of semantic clustering build on grammatical context found in (Bernard, 1997), but with important modifications, mainly due to our aim to develop a method as independant from specific languages as possible. So the stopword list and the stopword categories used in this paper were out of the picture. The left window (or any fixed-length window for that matter) was also too restrictive as we did not want to make any assumption as to order of parts of speech or type of syntax rules. Instead, we wanted to empirically discover recurrent patterns of very general words.

So the context we take into account is composed of (ordered) patterns of such words in the vicinity of a given word, inside sets that are delimited by punctuation markers. We called our method Grammatical Pattern Vector, or GraPaVec, though the relation of this algorithm to grammar is indirect (see subsection 4.2). GraPaVec has four steps:

- Trie preparation
- Pattern element selection
- Pattern discovery
- Word vectorization

4.1 Trie Preparation

We begin by importing every word in the corpus in a prefix Trie. A Trie is a structure that can represent a very large number of words in a format that is both

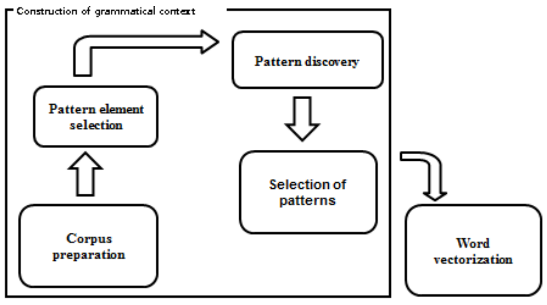


Figure 2: General view of GraPaVec.

economical and fast to explore[2]; it is more efficient here than hash-code or binary trees.

Each path from the root to a leaf of the Trie represents a word (see figure 3). Each node contains a unicode character. A node is marked as leaf the first time a word ends there, and its occurrences are incremented each time. Thus a node is a simple structure with a unicode character, a field indicating the number of occurrences (if not zero, the node is a leaf), pointers towards its sons and towards its brothers. A leaf can have sons, as words can be part of other words.

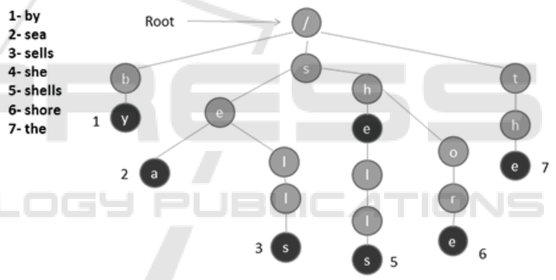


Figure 3: A Trie.

4.2 Pattern Element Selection

This is the most important step and the one where a human eye is necessary (for now). If the corpus is big enough, the most frequent words are markers with grammatical or very abstract function (with no independant meaning or referent, the *syncategoremes* of Aristotle); we tested this on English as well as on Arabic. The user – which just needs to know the language – has to set the frequency threshold that separates markers and "ordinary" words (lower frequency words). This is done by looking for the most frequent lexical item appearing in the list displayed by our system and establishing its frequency as threshold.

[2]Its maximal depth is given by the longest word in the corpus and its maximal breadth is given by the number of possible characters at any point. As shown by (Harris, 1968), in language the number of successors is constrained, so the tree quickly shrinks.

With the subcorpus used for evaluation (section 6), a threshold of 3,500 selected 155 markers. With the whole corpus, a threshold of 9,000 separated 196 markers. The whole corpus is about 980 times bigger than the evaluation corpus, the number of unique words is about 17.5 times bigger, but the number of markers is only 1,26 times bigger, and the threshold only 0.26 bigger. Thus there does not seem to be a clear relationship between the size of the corpus and this threshold.

Looking more closely, with the whole corpus, 53 markers were added, 12 were lost. Half of these 12 were combinations of markers that had correctly been classified; with better lemmatization, those would be eliminated, leaving an error margin of 3% relatively to the number of markers detected. On the whole, the biggest the corpus, the more homogeneous marker distribution is, and more neat its identification to grammatical words.

We compared these 196 words with the hand-made Arabic Stopwords Sourceforge resource: half of them (97) were not included in the resource (77 words on the 155 list). Most of them should have been included as stopwords; others were combinations of stopwords. More generally, though Arabic Stopwords includes 449 words, it seems rather incomplete, and could easily be enriched by our method.

4.3 Pattern Building

A pattern is a sequence of markers including sequences of ordinary words. For instance:

- the red book of Peter
- the car of George
- the heart of London

As *the* and *of* are higher frequency words, these phrases are instances of the same pattern: "the * of *". The star (joker) represents a sequence of ordinary words. Patterns are build according to the following principles (m represents a marker, x an ordinary word, p a punctuation):

- A pattern does not contain **p**.
- A pattern is a sequence of **m** and *.
- A pattern contains at least one *.
- * is a string of **x** with *n* as maximum length.
- * contains at least one **x**.

The maximum length *n* is called JokerLength. Let us take the following sequence, representing an extract from the corpus:

xmmxxmxxxpmxmmxxxxmpmmxxxm

Our objective is to generate all possible patterns compatible with this sequence. These patterns will be represented by sequences of **m** and *, as in <*mm*m>. Supposing that JokerLength = 3, we first obtain the following patterns:

- *mm*m* (followed by p)
- m*mm* (followed by more than 3 x)
- *m (followed by p)
- mm*m (end-of-file considered as p)

From each of these patterns all potential patterns included are deduced. For instance, <*mm*m*> contains the following sub-patterns:

- *m
- *mm
- *mm*
- *mm*m
- *mm*m*

Then we skip the first element and do the same with <mm*m*> and its subpatterns, and recurse until the pattern is finished. Of course, in real patterns, **m** is replaced by true markers; thus pattern <*m*m> is in reality a set of patterns differing by the nature of both '**m**'.

In the actual implementation, patterns are read from the corpus in a prefix Trie similar to the one used for words. Every star is a node that permits back reference from the ordinary word in the database to the positions it can occupy in the pattern Trie.

4.4 Word Vectorization

As the preceding process builds all possible patterns in the vicinity of a word, most of them will not be relevant and will not be repeated. We need a frequency threshold to eliminate spurious patterns that won't discriminate words.

We compute for each word the number of times it occurs in every selected pattern. This process yields a (sparse) matrix $\{\text{words} \times \text{patterns}\}$. We then eliminate from this matrix all patterns whose frequency is less than the pattern threshold selected.

Thus word vectorisation depends on three parameters: marker threshold, JokerLength and pattern threshold.

5 SELF ORGANIZING MAP

Self Organizing Map is an unsupervised neural network model designed by Kohonen (Kohonen, 1995).

In its standard version, it projects the space of input data on a two-dimension map. It implicitly does a dual clustering: on one hand, the data is clustered into neurons, and on the other hand the clusters themselves are grouped by similarity in the map. Its operation is in "best matching unit" mode: all neurons compete for each input vector and the best matching neuron is the winner.

X being the input vector, j an index on the n neurons in the map, W_j the memory vector of the neuron j, the winner, $j*$, is determined by equation 1, where $d(x,y)$ is a distance measure:

$$d(X, W_{j^*}) = \min_{j \in \{1-n\}} d(X, W_j) \quad (1)$$

The distance can be euclidian (usual value), Manhattan, or some other. It can be replaced with a similarity measure as cosine (normalized dot product, eq. 2), if *min* is replaced by *max* in eq. 1. With sparse vectors cosine similarity drastically reduce computation time.

$$CosSim(X, W_j) = \frac{\sum_{i=1}^{n} X_i W_{i,j}}{\parallel X \parallel \times \parallel W_j \parallel} \quad (2)$$

Every neuron has a weigth vector W_j of the dimension of the input vector, initialized randomly and maybe pre-tuned to the set of possible values. In the learning phase the winner and every neuron in its neighbourhood learn the input vector, according to eq. 3, where $N_\sigma(i,j)$ is the neighbourhood in radius σ; the bracketed superscript indicates the epoch.

$$W_j^{(t+1)} = W_j^{(t)} + \alpha^{(t)} N_\sigma^{(t)}(j, j^*)(X_i^{(t)} - W_j^{(t)}) \quad (3)$$

The learning rate α decrease in time following equation 4, where $\alpha^{(0)}$ is its initial value.

$$\alpha^{(t)} = \alpha^{(0)}(1 - \frac{t}{t_{max}}) \quad (4)$$

Learning in the neighbourhood of the winner decrease in space following here the gaussian in equation 5, which yields better results than mexican hat or other variants. $M(i,j)$ is the Manhattan distance between indexes.

$$N_\sigma^{(t)}(j, j^*) = e^{-\frac{M(j,j^*)}{2\sigma^{2(t)}}} \quad (5)$$

σ obeys equation 6, where $\sigma^{(0)}$ is the radius initial value, typically the radius of the map, and $\sigma^{(t_max)}$ is its final value, typically 1.

$$\sigma^{(t)} = \sigma^{(0)}(\frac{\sigma^{(0)}}{\sigma_{t_{max}}})^{\frac{t}{t_{max}}} \quad (6)$$

Our implementation gives the choice of euclidian distance, Manhattan, cosine similarity; different topologies for the neighbourhood (square or hexagonal), initialize memory to the center of learning set values or randomly.

6 EVALUATION

As our final objective was to produce new synsets, we wanted to check whether AWN existing synsets were correctly retrieved, that is, whether the words of a synset were all clustered together. At first we thoroughly assessed the quality of the 11,269 existing AWN 2.1 synsets. This study yielded the following issues:

A) 4,712 synsets are singletons.

B) 1,110 are subsets of others.

C) A non-negligible number of synsets are false.

Type (A) synsets would have artificially increased the recall value of any method (they would always be in the same cluster). As synsets of type (B) do not form a complete partition of their supersets, some words would not have been taken into account and the number of singletons would have increased. After eliminating these synsets, we were left with 5,807 synsets. It is easy to see why type (C) synsets were not to be used, but much less easy to eliminate them, as it has to be done by hand. For our experiments, we controled and choose 900 synsets grouping 2,107 words. Those synsets can be found at https://sites.google.com/site/georgeslebboss.

The evaluation corpus contained the documents of our large corpus containing at least one of the words of these synsets. We ended up with an evaluation corpus of 7,787,525 words and 395,014 unique words. Quality of evaluation will increase as AWN itself increases in quality.

In order to compute the F-score (harmonic mean of recall and precision) of the four methods tested here, we run them on the evaluation corpus, insert result in database, then cluster vectors obtained from each method with SOM model. The resulting clusters are compared to our synsets: we count the number of synsets whose at least two words are clustered together. Let us call C this number, T the total number of synsets, and S SOM number of output cells; recall is computed as $\frac{C}{T}$ and precision is computed as $\frac{C}{S}$.

We tuned parameters to their best values separately for each method with many tests.

- Best values for common parameters
  1. orthographic normalization: true
  2. lemmatization: true
  3. SOM topology: hexagonal

4. SOM measure: cosine similarity
5. SOM mapsize: 35 **x** 26

- Best values for GraPaVec parameters

1. marker threshold: 3500 in evaluation corpus, 9000 in the large one
2. JokerLength: 4
3. pattern threshold: 300
   These parameter values yield the following vector dimensions: 1,571 in evaluation corpus, 1,869 in the large one.

- Best values for Word2Vec (Skipgram and CBOW) and Glove parameters

Table 2: Word2Vec and Glove Parameter Values.

	Skipgram	CBOW	GloVe
Vector dim.	300	300	50
Window size	10	7	15
Sample	1e-3	5	N/A
Hier. softmax	0	0	N/A
Negative samp.	10	5	N/A
Iterations	10	5	15
Min count	5	5	5
Learning rate	0.025	0.05	N/A
X Max	N/A	N/A	10

Normalization and lemmatization were the parameters that, along with the SOM map size most influenced the results. Normalizing increments the F-score of any method by 12%; lemmatizing increments it by 17%; adding both increments it by 28%. The table 3 shows F-scores with raw, normalized, lemmatized and both (normalized and lemmatized) corpus.

Table 3: Effect of preprocessing on F-scores.

Corpus	GraPaVec	Skipgram	CBOW	GloVe
raw	54.1	27.2	24.3	28.6
norm.	65.5	38.5	35.6	40.0
lemm.	70.7	43.7	40.8	45.1
both	82.1	55.1	52.2	56.6

When SOM mapsize is 10% bigger than the number of synsets to be retrieved, it downgrades the result of more than 20% with all three methods; it is also the case when the map size is smaller, which is expected, but the effect is less drastic. The table 4 shows some results (all with lemmatization and normalization); we have indicated also the results with the chosen value (35 *x* 26), corresponding to the number of synsets.

Table 4: Effect of map sizes on F-scores.

Mapsize	GraPaVec	Skipgram	GloVe
34 *x* 24	80.72	54.82	56.17
Chosen	82.1	55.1	56.6
35 *x* 27	79.78	52.47	53.98
35 *x* 28	77.23	51.81	52.23
35 *x* 30	61.54	41.44	42.05

The final evaluation using the best parameters for each method yields a F-score of 82.1% for GrapaVec, 55.1% for Word2Vec's Skipgram, 52.2% for CBOW and 56.6% for Glove.

7 CONCLUSION AND DISCUSSION

To be completely open, we did not expect such gap between the results of GraPaVec and the other methods. We looked for biases in our procedure. The only parameters that are common to all methods are those of SOM (topology, distance and map size). The map size mostly depends on the number of clusters to be found and we had to be close to the number of synsets. The choice of hexagonal topology, recommended by Kohonen himself on general grounds, gave the best results for all methods. Cosine similarity has no reason to favor one method. That leaves SOM choice, but there is no clear reason why it should biase in favor of GraPaVec.

Could sparsity be a factor? The below results show vector sparsity in the evaluation corpus; while GraPaVec vectors are indeed six times sparser than Word2Vec's, these are in turn six times sparser than GloVe's, with no noticeable effect on the results:

- GraPaVec : 0.27 %
- Skipgram : 1.65 %
- CBOW : 1.65 %
- GloVe : 9.89 %

One bias is clear, though: GraPaVec is twice as time consuming as Skipgram, the most time consuming of the other methods, as shown by the below results in minutes (using Laptop Core i5 with 8 GB RAM); pattern construction consumes 4/5 of GraPaVec time:

Method	big corpus	evaluation corpus
GraPaVec	483	125
Skipgram	242	63
Glove	66	16
CBOW	48	12

But this does not explain the F-score gap.

Perhaps the solution is to look at what Levy et al. (Levy et al., 2015) call hyperparameters. Here the type of context could have played the major role. It would be interesting to twist Word2Vec and Glove to apply them to such contexts. Another element could have played some role: the corpus itself and the language under study. As Goldberg (Goldberg, 2014) puts it,

> It is well known that the choice of corpora and contexts can have a much stronger effect on the final accuracy than the details of the machine-learning algorithm being used [...]

Either way we achieved here two aims: building Arabic word clusters on the basis of Arabic corpora, a first step in enriching AWN, and showing that patterns of higher frequency words, mostly grammatical words, thrown away as "empty words" by most methods, are operative in semantic lexical clustering at least in Arabic. More work on the contexts is needed here.

There are still a number of questions to be adressed. Is it possible to automatize the selection marker threshold? What impact on the results would have moving this threshold down or up? Reducing the computational cost of GraPaVec is a must in order to be able to do more extensive tests and is one of our first objectives for now.

In a near future we also aim to produce synsets based on our work; to try our hand at other languages, in order to see if those results are language specific, and to use a dynamic growing neural model that can find by itself the number of categories.

REFERENCES

Abdelali, B. and Tlili-Guiassa, Y. (2013). Extraction des relations sémantiques à partir du Wiktionnaire arabe. *Revue RIST*, 20(2):47–56.

Abdulhay, A. (2012). *Constitution d'une ressource sémantique arabe à partir d'un corpus multilingue aligné*. PhD thesis, Université de Grenoble.

Abouenour, L., Bouzoubaa, K., and Rosso, P. (2008). Improving Q/A using Arabic WordNet. In *Proc. of the 2008 International Arab Conference on Information Technology (ACIT'2008), Tunisia*.

Abouenour, L., Bouzoubaa, K., and Rosso, P. (2010). Using the Yago ontology as a resource for the enrichment of named entities in Arabic WordNet. In *Proceedings of The 7th International Conference on Language Resources and Evaluation (LREC 2010) Workshop on Language Resources and Human Language Technology for Semitic Languages*, pages 27–31.

Abouenour, L., Bouzoubaa, K., and Rosso, P. (2013). On the evaluation and improvement of Arabic WordNet coverage and usability. *Lang Resources & Evaluation*, 47(3):891–917.

Al-Barhamtoshy, H. M. and Al-Jideebi, W. H. (2009). Designing and implementing Arabic WordNet semantic-based. In *the 9th Conference on Language Engineering*, pages 23–24.

Al Hajjar, A. E. S. (2010). *Extraction et gestion de l'information à partir des documents arabes*. PhD thesis, Paris 8 University.

Alkhalifa, M. and Rodriguez, H. (2008). Automatically extending named entities coverage of Arabic WordNet using Wikipedia. *International Journal on Information and Communication Technologies*, 1(1):1–17.

Bernard, G. (1997). Experiments on distributional categorization of lexical items with Self Organizing Maps. In *International Workshop on Self Organizing Maps WSOM'97*, pages 304–309.

Black, W., Elkateb, S., Rodriguez, H., Alkhalifa, M., Vossen, P., Pease, A., and Fellbaum, C. (2006). Introducing the Arabic WordNet project. In Sojka, Choi, F. and Vossen, editors, *In Proceedings of the third International WordNet Conference*, pages 295–300.

Goldberg, Y. (2014). On the importance of comparing apples to apples: a case study using the GloVe model. Google docs.

Hajjar, M., Al Hajjar, A. E. S., Abdel Nabi, Z., and Lebboss, G. (2013). Semantic enrichment of the iSPEDAL corpus. In *3rd World Conference on Innovation and Computer Science (INSODE)*.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Harris, Z. S. (1968). *Mathematical structures of language*. John Wiley & Sons.

Honkela, T., Kaski, T., Lagus, K., and Kohonen, T. (1997). WEBSOM–Self-Organizing Maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland*, pages 310–315. Helsinki University of Technology.

Khoja, S., Garside, R., and Knowles, G. (2001). An Arabic tagset for the morphosyntactic tagging of Arabic. *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, 13:341–350.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer, Berlin.

Lebboss, G. (2016). *Contribution à l'analyse sémantique des textes arabes*. PhD thesis, University Paris 8.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Lund, K., Burgess, C., and Atchley, R. A. (1995). Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th annual conference of the Cognitive Science Society*, volume 17, pages 660–665.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representation, Workshop Track*, page 1301.

Miller, G. A. (1995). Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Miller, G. A., Fellbaum, C., Tengi, R., Wolff, S., Wakefield, P., Langone, H., and Haskell, B. (2005). *WordNet 2.1*. Cognitive Science Laboratory, Princeton University.

Niles, I. and Pease, A. (2003). Linking lexicons and ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE 03), Las Vegas, Nevada*, volume 2, pages 412–416, Las Vegas, Nevada, USA.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors forword representation. In *EMNLP*, volume 14, pages 1532–1543.

Raafat, H., Zahran, M., and Rashwan, M. (2013). Arabase: A database combining different arabic resources with lexical and semantic information. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, pages 233 – 240. Scitepress.

Regragui, Y., Abouenour, L., Krieche, F., Bouzoubaa, K., and Rosso, P. (2016). Arabic WordNet: New content and new applications. In *Proceedings of the Eighth Global WordNet Conference*, pages 330–338, Bucharest, Romania.

Rodriguez, H., Farwell, D., Farreres, J., Bertran, M., Alkhalifa, M., Martí, M. A., Black, W., Elkateb, S., Kirk, J., Pease, A., Vossen, P., and Fellbaum, C. (2008). Arabic WordNet: Current state and future extensions. In *Proceedings of The Fourth Global WordNet Conference, Szeged, Hungary*, number 387–405, Marrakech (Morocco).

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Vossen, P. (2004). EuroWordNet: a multilingual database of autonomous and language-specific WordNets connected via an Inter-Lingual Index. *International Journal of Lexicography*, 17(2):161–173.