Wan2Vec: Embeddings Learned on Word Association Norms

Gemma Bel-Enguix [a], Helena Gómez-Adorno [b,*], Jorge Reyes-Magaña [a,c], and Gerardo Sierra [a]
[a] *Instituto de Ingeniería (II), Universidad Nacional Autónoma de México (UNAM), Mexico City, Mexico*
E-mails: gbele@iingen.unam.mx, gsierram@iingen.unam.mx
[b] *Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS), Universidad Nacional Autónoma de México (UNAM), Mexico City, Mexico*
E-mail: helena.gomez@iimas.unam.mx
[c] *Facultad de Matemáticas (FM), Universidad Autónoma de Yucatán (UAdY), Merida-Yucatan, Mexico*
E-mail: jorge.reyes@correo.uady.mx

Abstract. Word embeddings are powerful for many tasks in natural language processing. In this work, we learn word embeddings using weighted graphs from word association norms (WAN) with the node2vec algorithm. Although building WAN is a difficult and time-consuming task, training the vectors from these resources is a fast and efficient process. This allows us to obtain good quality word embeddings from small corpora. We evaluate our word vectors in two ways: intrinsic and extrinsic. The intrinsic evaluation was performed with several word similarity benchmarks, *WordSim-353*, *MC30*, *MTurk-287*, *MEN-TR-3k*, *SimLex-999*, *MTurk-771* and *RG-65*, and different similarity measures achieving better results than those obtained with word2vec, GloVe, and fastText, trained on a huge corpus. The extrinsic evaluation was done by measuring the quality of sentence embeddings using transfer tasks: sentiment analysis, paraphrase detection, natural language inference, and semantic textual similarity.

Keywords: Word Association Norms, Word Embeddings, Learning

1. Introduction

Semantic representation of words in a vector space has been an active research field over the past decades. Computational models like singular value decomposition (SVD) and latent semantic analysis (LSA) [1] are able to model continuous representations of words (embeddings) from term-document matrices. Both methods are able to reduce an n-dimensional term-document matrix using only the most important dimensions.

More recently, *Mikolov et al.* [2] introduced word2vec, inspired by the distributional hypothesis, which states that words in similar contexts tend to have similar meanings [3]. This technique uses a neural network to learn lexical vector representations by predicting other words in the neighborhood. Distributed vector representations obtained with word2vec have the capability to preserve linear regularities between words.

Some alternative methods have been designed to improve the performance of word2vec. GloVe [4] aims to be an efficient vector model by training nonzero elements in a word-word co-occurrence matrix. FastText [5] is an approach based on the skip-gram model, with the difference that in this method every word is represented as a bag of character n-grams. *Mikolov et al.* [6] claim that the models trained with fastText exhibit the best degree of accuracy compared to other systems, becoming the new state of the art in distributed representations of words.

In order to build a suitable and reliable semantic vector space model able to capture semantic similarities and linear regularities of words, huge volumes of text are necessary. Although methods for learning word vectors are fast and efficient to train, and pre-trained embeddings are usually available online, it is still computationally expensive to train huge volumes

[*]Corresponding author. E-mail: helena.gomez@iimas.unam.mx.

of data in non-commercial environments, i.e., personal computers. Acquisition of semantic representations require large handcrafted knowledge bases or huge volumes of text to be exploited. Some word2vec embeddings are trained on billions of words, which implies a computational cost that is not very realistic from the cognitive point of view either. In addition to this, only really big sources like Wikipedia, an encyclopedia, can provide the number of lexical items necessary to perform such processes. Encyclopedias have plenty of terms that are not part of common language (ie., Latin names for plants and animals, geographical names, etc.), introducing some unneeded noise to the vectors.

Even though millions of different words can be computed for languages like English, this is not the actual size of the vocabulary an English speaker uses in daily life. Psycholinguists do not agree on the number of words that comprise the core of a language. In the thirties of the 20th Century, Ogden [7] suggested that there is a small set of 850 items that constitute what he calls "basic English". West [8], elaborated a list with the 2000 more frequent words in English that were extracted from a corpus. The resource, called General Service List (GSL), was oriented to the use of ESL learners and teachers. This was updated in 2013 by Browne et al. [9], who published the New General Service List (NGSL). The authors tested both lists in the Cambridge English Corpus (CEC), concluding that the GSL covered 84% and the NGSL 92% of the words of the resource [10].

Leaving aside the field of English as a Second Language, the theory explained above has been broadly criticized [11]. However, the vocabulary of daily life is limited compared to encyclopedic resources, and the idea to restrict to the most common words is still followed in several publications of lexicography. As an example, the Longman Dictionary of Contemporary English (LDOCE) [12] uses a list of the two thousand most used words in English to define the lemmas of the whole dictionary, 230.000 in the 2003 edition[1].

Summing up, it seems that having lexical resources with a more restricted number of words is not necessarily a limitation, although the ideal coverage of the vocabulary to balance complexity and utility is controversial.

In another vein, methods based on co-occurrences of words, like word2vec, provide a good idea on the syntagmatic and paradigmatic relations of words, but fail to show their psychological and associative connections. When envisioning some writing assistants and applications to help with real problems of dysnomia, these latter types of relations cannot be dismissed.

Keeping the previous ideas in mind, it makes sense to look for models that can work with more restricted sets of both tokens and types. Somehow, these methods can decrease the recall, but the computational complexity and the efficiency will be increased. Moreover, it is highly recommended to consider corpora that can provide other information on how the lexicon is structured and accessed, rather than corpora where the relation between words are only based on co-occurrence. Wordnet [13], semantic networks or word association networks are possible sources for these experiments. In this paper, we evaluate the convenience of using word association norms as a basis for learning relations between words.

Word association (WA) tests are an experimental technique for discovering the way that human minds structure knowledge [14]. In free association experiments, a person is asked to respond with the first word that comes to mind in response to a given *stimulus* word. Free WA tests are able to produce rich types of associations that can reflect both semantic and episodic memory contents [15] in the form of general word association norms (WAN).

The goal of this paper is to learn continuous feature representations for nodes (that represent words) in WAN. Our hypothesis is that word embeddings learned from WAN map the organized lexical items arranged in the lexicon to a vector space, thus learning richer representations. *Grover and Leskovec* [16] introduced an algorithm called node2vec that is able to learn mappings of nodes to a continuous vector space taking into account the network neighborhoods of nodes. The algorithm performs a biased random walk to explore different neighborhoods in order to capture not only the structural roles of nodes (words) in the network but also the communities they belong to. Our proposal is called *wan2vec*, because vectors are learned from a network built upon a WAN, by means of the node2vec algorithm.

The rest of the paper is organized as follows. Section 2, discusses some related work. In Section 3, we present a well-known compilation of word association norms, the Edinburgh Association Thesaurus, and we describe the methodological framework for learning word embeddings from WAN. In Section 4, we present the evaluation of the generated vectors using standard

[1]http://www.pu-kumamoto.ac.jp/~rlavin/resources/wordlists/LongmanDefiningVocab.html

data sets for word similarity in English. In section 5, an extrinsic evaluation is conducted, testing the performance of wan2vec with other tasks related to NLP. Finally, in Section 6 we draw some conclusions and point to possible directions of future work.

2. Related Work

The idea of word associations was first proposed by psychologists to tackle disorders like dementia or aphasia. Within cognitive psychology, *Collins* [17] applied them to simulate memory processes. From psycho-linguistics, *Clark* [18] presents free associations as an ability that can reveal some properties of the mechanisms of language.

Linguistics and Psycholinguistics used semantic networks [19], which are graphs relating words [20], not only to study the organization of the vocabulary, but also to approach the structure of knowledge.

Word association norms are corpora of free association words. One of the first examples is provided by *Kent & Rosanoff* [21], who used this method for comparisons of words, introducing 100 emotionally neutral test words. They conducted the first large scale study with 1000 test subjects, and concluded that there was uniformity in the organization of associations and that people shared stable networks of connections among words [22].

Many languages have corpora of WAN. In the past decades, some other association lists were elaborated with the collaboration of a large number of volunteers. In recent years, the web has become the natural way to get data to build such resources. *Jeux de Mots*[2] provides an example in French [23], whereas the *Small World of Words*[3] contained datasets in 14 languages at the time of writing. The repositories that are collected online have certain limitations, especially the lack of control over who is actually playing, the linguistic proficiency of the users, and their age, gender or level of studies.

Among the best-known resources available on the web for English are the Edinburgh Associative Thesaurus[4] (EAT) [24] and the compilation of *Nelson et al.* [25][5].

[2]http://www.jeuxdemots.org/

[3] https://smallworldofwords.org/.

[4]http://rali.iro.umontreal.ca/rali/?q=en/Textual%20Resources/EAT

[5]http://web.usf.edu/FreeAssociation

Borge-Holthoefer & Arenas [15] introduced a Random Inheritance Model (RIM) for extracting semantic similarity relations from free association information. The obtained vectors were compared with LSA-based vector representations and the WAS (word association space) model. Their results indicate that RIM can successfully extract word feature vectors from a free association network.

In recent years, *Bel-Enguix et al.* [26] used graph analysis techniques to compute associations from large texts collections. Furthermore, *Garimella et al.* [27] introduced a demographic-aware word association model based on a neural net skip-gram architecture, outperforming generic methods for computing associations that do not take into account writer's demographics.

Sinopalnikova and Smrz [28] presented a methodological framework for building and extending semantic networks with word association thesaurus (WAT), including a comparison of quality and information provided by WAT vs. other language resources. The authors showed that WATs are comparable to balanced text corpora and can replace them in case of absence of a corpus.

In a recent work by *De Deyne et al.* [29] the authors introduced a spreading activation approach in order to encode a semantic structure from a word association network, specifically part of the *Small World of Words*. The word association-based model was compared with a word embeddings model (word2vec) using relatedness and similarity judgments from humans, obtaining an average of 13% of improvement over the word2vec model.

The task of measuring the semantic similarity and relatedness has been a key problem in NLP, with multiple implications in lexical semantics [30]. To approach the problem, some authors [31–35] have used pre-existing knowledge resources like Wordnet [13]. Other approaches are based on distributional semantics. The vector-based methods, which calculate the semantic realatedness by means of the extraction of characteristic vectors for words, typically use the cosine similarity measure [36, 37]. Recently, some methods have been introduced that use different algorithms for obtaining the similarity between two given vectors; among them APSyn [38, 39], that ranks the most associated contexts of two words and computes the extent of the intersection between them.

In this work, we aim at learning vector representation of words using as a basic resource the EAT [24], a compilation of word association norms in English. We claim that, in spite of the limited vocabulary, word associations capture more efficiently lexical connections and imply a lower complexity in both processing time and space. *De Deyne et al.* developed a similar idea in [29], but unlike their work, we use the complete WAN corpus (not only the largest connected component), increasing in this way the coverage of the vocabulary by 5% and maintaining high quality vectors.

3. Learning embeddings on the Edinburgh Association Thesaurus WAN

The EAT was mainly collected with undergraduate students from different British universities. The participants were age between 17 and 22, among which 64% were males and 36% were females. Every informant gave responses for 100 words, and every word was given to 100 participants. The resource was elaborated between 1968 and 1971, and published in 1973. Although it could be said that EAT presents an "old stage" of the language, it is the most complete and balanced WAN elaborated so far.

We used an XML version of the resource[6], prepared by the University of Montreal, that consists of 8,211 stimulus words, and 20,445 different words including both, stimuli and responses, being these the nodes of the graph.

The graph representation of the EAT corpus is formally defined by $G = \{V, E, \phi\}$ where:

- $V = \{v_i | i = 1, ..., n\}$ is a finite set of nodes of length n, $V \neq \emptyset$, corresponding to the *stimulus* and its *associates*.
- $E = \{(v_i, v_j) | v_i, v_j \in V, 1 \leqslant i, j \leqslant n\}$, is the set of edges.
- $\phi : E \rightarrow \mathbb{R}$, is a weight function over the edges.

The graph is undirected, so that every stimulus is connected to every associated word without any precedence order.

Two different functions (ϕ) were used to assign weight to the edges: Frequency and Association Strength [40]. Only the former is included in the data provided by the EAT. The latter was calculated from the results of frequency for every word associated to its stimulus.

Frequency (ϕ_F) Counts the number of occurrences of every *response* associated to its *stimulus* in the whole corpus. In order to use this information, we calculated the inverse frequency (ϕ_{IF}), that is defined in the following way: being ϕ_F the frequency of a given associated word, and $\Sigma\phi_F$ the sum of the frequencies of the words connected to the same *stimulus*, $\phi_{IF} = \Sigma\phi_F - \phi_F$.

Association Strength (ϕ_{AS}) Establishes a relation between the frequency (ϕ_F) and the number of associations for every stimulus. It is calculated as follows: being ϕ_F the frequency of a given associated word, and $\Sigma\phi_F$ the sum of the frequencies of the *responses* to the same *stimulus* (the total number of responses), the association strength (ϕ_{AS}) of the word to such *stimulus* is given by the formula:

$$\phi_{AS} = \frac{\phi_F}{\Sigma\phi_F}$$

For our experiments, we used the inverse association strength, ϕ_{IAS} in order to prepare the system to work with graph-based algorithms:

$$\phi_{IAS} = 1 - \frac{\phi_F}{\Sigma\phi_F}$$

Note that in this work we use ϕ_{IF} and ϕ_{IAS} because node2vec traverse the shorter paths, or those with a smaller weight, whereas in WAN the higher results in Frequency and Association Strength are denoted with higher scores. In order to have a compatible quantification, the concepts 'Inverse Frequency' and 'Inverse Association Strength' have been introduced.

3.1. Node2vec

The node2vec algorithm [16] finds a mapping $f : V \rightarrow \mathbb{R}^d$ transforming the nodes of a graph into vectors of d-dimensions. It defines a network neighborhood $N_s(u) \subset V$ for every node $u \in V$ through a sampling strategy S. The aim of the algorithm is to maximize the probability of observing subsequent nodes in a fixed length random walk.

The sampling strategy designed in node2vec enables to explore neighborhoods with flexible biased-random walks. Parameters p and q control the switching between breath-first (BFS) and depth-first (DFS) graph searches. Parameter p controls the likelihood of immediately revisiting a node in the walk and parameter q allows the search to differentiate between "in-

[6]http://rali.iro.umontreal.ca/rali/?q=en/Textual%20Resources/EAT

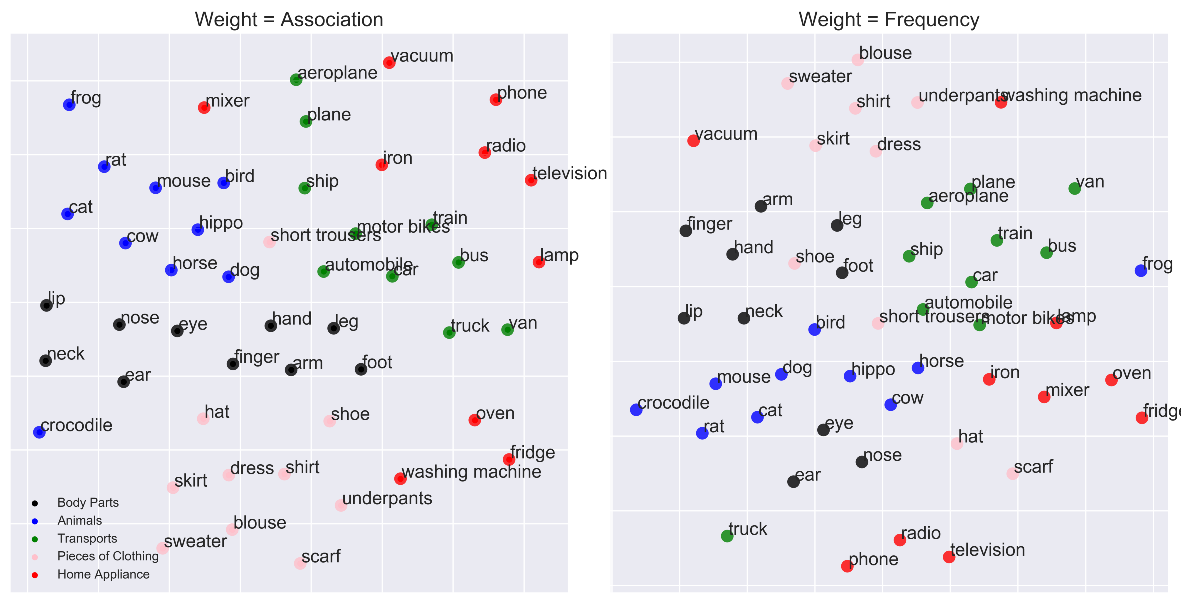

Fig. 1. Projection of word vectors for 5 semantic groups (ten words each). Colors are codified as follows: animals - red, transportation - black, body parts - blue, household appliances - green, clothes - pink.

ward" and "outward" nodes. The values used in this work were $p = 1$ and $q = 1$. Thus, choosing the right balance allows to preserve both community structure and structural equivalence between nodes in the new vector space.

In this work, we used the implementation available on the website[7] of the node2vec project with default values for all parameters. We examined the quality of the vectors with different number of features d and different number of walk lengths l.

4. Word Embeddings Evaluation

There are several evaluation methods for unsupervised embedding techniques [41], categorized as intrinsic and extrinsic. The extrinsic evaluation (that will be tested in section 5) consists in evaluating the quality of the word vectors in Natural Language Processing (NLP) tasks [42, 43]. The improvement is measured in the performance metric specific to the evaluated task. Intrinsic evaluations test the ability of word vectors to capture syntactic or semantic relationships [44] of words. The hypothesis of the intrinsic evaluation states that similar words should have similar representations. This section is devoted to perform the intrinsic evaluation of *wan2vec*.

Thus, to evaluate similarity we first performed a visualization of a sample of words using a T-SNE[8] projection of word embeddings in a two-dimensional space. The experiment was carried out using both the ϕ_F and the ϕ_{AS} as weight functions, note that for the visualization experiments we did not used the inverse version of the weighting functions. We examined 50 words, divided into 5 semantic groups. Figure 1 shows how related words are clustered. When we used the ϕ_{AS} as the graph weighting function, the method mostly gathers the elements of the 5 semantic groups. It can be observed that the animals and body parts are clustered together. On the other hand, 'artifacts', like means of transportation or household appliances are grouped too. Clothing has its own space, except for *short trousers*, which is isolated. An interesting phenomenon is that *washing machine* is closer to pieces of clothing, and it could really belong to two semantic groups: household appliances and clothing.

When the ϕ_F works as weighting function, the borders of the clustering groups are not so well defined. *Washing machine* has still a location close to some pieces of cloth, but this category is spread without or-

[7]http://snap.stanford.edu/node2vec/

[8]http://scikit-learn.org/stable/modules/manifold.html#t-sne

der in the two-dimensional space. *Truck* and *frog* are also far from their expected group.

4.1. Similarity and relatedness

In addition, we evaluated the ability of *wan2vec* to capture semantic relations through a similarity task. We used two subsets of the *WordSim-353* [45] benchmark comprised of 353 semantically related term pairs with similarity scores given by humans. This list does not distinguish between the concepts of similarity and relatedness. *Aguirre et al.* [30] split the list into two different sets, one with relatedness scores, containing 252 pairs [EN-WS-353-REL], and another with 203 pairs linked by similarity [EN-WS-353-SIM]. There is an overlap between the lists, all of the term pairs belonging to the 353 pairs of *WordSim 353*.

The *WordSim-353* is based on works from the nineties elaborated in the USA. Because of the time and geographical differences, several words from this list are not included in the EAT, a British collection from the early seventies. This fact is not due to a lack of expressivity in WAN, but it is caused because some objects, people and ideas did not exist when the EAT was collected, or they were used in very different contexts. An example can be the word *hardware*, a neologism from the nineties, or *Maradona*, the soccer star from the eighties. As a result, 183 pairs in the list of similarity are in the WAN corpus, while 214 out of 253 are in the relatedness compilation. To deal with the non-inclusion of every word of the testing data sets in our EAT WAN, we introduced the concept of overlap in the experiments and calculated the total number of common words between the lists that are being compared. The others are excluded from the evaluation. In principle, having large overlaps is a positive feature for *wan2vec*.

We have also tested our method with SimLex-999 [SimLex-999] [46], a resource that contains 999 pairs of words and explicitly quantifies similarity in a way that pairs related by association or relatedness have a low rating. The overlap between the EAT and SimLex-999 is 968, which means that almost every word in the test data is covered by *wan2vec*.

The rest of the data sets we used for our experiments, measure the relatedness of words more than the similarity.

The Amazon Mechanical Turk data [MTurk-287] [47] consists of 287 word pairs, elaborated with the collaboration of the Amazon's Mechanical Turk workers. The overlap with EAT is 203.

MEN [MEN-TR-3k] [48] is an evaluation benchmark that contains 3,000 pairs of randomly selected words. Each pair is scored on a [0, 1] normalized semantic relatedness scale via ratings obtained by crowdsourcing on the Amazon Mechanical Turk. We have a very high overlap with this corpus, 2,727 out of 3,000 word pairs.

The MTURK-771 data set [MTurk-771] [49] evaluates the relatedness between 771 word pairs. It was obtained with the evaluation of Amazon's Mechanical Turk workers. EAT covers 698 word pairs of this data set.

The RG65 [RG-65][50] collected data for 65 pairs of non-technical words. Their objective was to evaluate the judgments and perceptions about synonymy, and because of these, they used different pairs, in a range from highly synonymous to totally unrelated words. Every word pair in the RG65 is also in the EAT.

Finally, we evaluated the *wan2vec* embeddings with the MC-30 [MC-30] [13] benchmark. This list contains 30 word pairs, all of them included in the EAT.

Table 1 reports the Spearman rank order cosine correlations between the previously described datasets and the word embeddings of different dimensions learned on undirected graphs of the WAN EAT, being the weight of the edges given by the ϕ_{IF}. Table 2 shows the performance based on cosine too, of the *wan2vec* embeddings similarity and relatedness predictions using the ϕ_{IAS} as weighting function. For each table, the total number of word pairs (n) and the number of word pair overlap (n (Overlap)) are given.

From these tables it can be seen that, in general, the *wan2vec* embeddings learned on the graph weighted with the ϕ_{IAS} weighting function obtained higher correlation with human judgments than the embeddings learned on the graph weighted with the ϕ_{IF} function. This was expected because, generally, the association strength is a normalization of frequency that should give a more general idea on the relationship between two words. However, and for the same reason, the results are not significantly different between both weighting methods. Summing up, ϕ_{IAS} could be considered the measure of the weight of edges of the graphs by default, although in a simple approach ϕ_{IF} would work as well without a great loss of expressiveness.

As we have stated before, we are testing only the words in the intersection between the test lists and the WAN. The column n in tables 1 and 2 corresponds to the number of pairs in the benchmarks, while the third column, $n(Overlap)$ shows the number of items that

Benchmark	***n***	***n (Overlap)***	**wan2vec dimensions**						
			25	50	100	128	200	300	1000
MC-30	30	30	0.6639	0.8244	0.7817	0.8117	0.7814	**0.8500**	0.7570
WS-353-SIM	204	183	0.6823	0.7637	**0.7839**	0.7481	0.7674	0.7448	0.7276
MTurk-287	287	203	**0.7278**	0.7089	0.6989	0.6834	0.6701	0.6476	0.5626
WS-353-REL	253	214	0.5987	**0.6953**	0.6926	0.6661	0.6855	0.6455	0.5816
MEN-TR-3k	3000	2720	0.7836	0.8000	**0.8031**	0.7947	0.7910	0.7629	0.7089
SimLex-999	999	968	0.4297	0.4548	0.4959	0.5072	**0.5196**	0.5127	0.5052
MTurk-771	771	698	0.6795	0.7083	**0.7315**	0.7170	0.7228	0.7007	0.6674
RG-65	65	65	0.7064	**0.8167**	0.7686	0.8058	0.7605	0.7973	0.7106
Average			0.6590	**0.7215**	0.7195	0.7168	0.7123	0.7077	0.6526

Table 1

Spearman rank order correlations between wan2vec predictions (based on cosine similarity) and the evaluation benchmarks. The WAN graph was built using the ϕ_{IF} as weighting function.

Benchmark	***n***	***n (Overlap)***	**wan2Vec dimensions**						
			25	50	100	128	200	300	1000
MC-30	30	30	0.6493	0.7725	0.7921	0.8206	0.8480	**0.8582**	0.7409
WS-353-SIM	204	183	0.6956	0.7517	0.7770	0.7633	**0.7775**	0.7623	0.6843
MTurk-287	287	203	**0.7270**	0.7168	0.6976	0.6920	0.6610	0.6661	0.5697
WS-353-REL	253	214	0.6228	0.6967	**0.7051**	0.6854	0.6984	0.6586	0.5711
MEN-TR-3k	3000	2720	0.7825	0.8030	**0.8075**	0.7992	0.7857	0.7646	0.7140
SimLex-999	999	968	0.4206	0.4614	0.4909	0.5009	0.5101	**0.5103**	0.5035
MTurk-771	771	698	0.6769	0.7122	0.7303	**0.7364**	0.7196	0.6973	0.6637
RG-65	65	65	0.7506	**0.8213**	0.7982	0.7850	0.7889	0.7913	0.7153
Average			0.6657	0.7169	**0.7249**	0.7228	0.7230	0.7136	0.6453

Table 2

Spearman rank order correlations between wan2vec predictions (based on cosine similarity) and the evaluation benchmarks. The WAN graph was built using the ϕ_{IAS} as weighting function.

can be found also in the WAN. It can also be that there is a significant overlap between the word pairs of each of the data sets and the vocabulary of our *wan2vec* embeddings. This means that, in general, despite the small size of the resource (only 20,445 nodes) the model has a high level of expressiveness, covering most of the words included in the lists.

Concerning the dimensions of the vectors, 50 and 100 proved to be the most efficient. With ϕ_{IF}, vectors of dimension 50 and 1000 have better performance than with ϕ_{IAS}. However, this last weighting function reaches better results with the rest. Node2vec is set by default to dimension 128, but for *wan2vec* smaller vectors seem to have a better performance. It is not clear if this can be caused by the relatively small number of nodes of our graph. When using the ϕ_{IF} weighting function, the 50-dimensional vectors have a higher average Spearman rank correlation, although the vectors with 100 dimensions obtained the best results on three of the corpora. On the other hand, the *wan2vec* embeddings learned on the graph weighted with the ϕ_{IAS} weighting function, vectors with 100 dimensions seem to be enough in average and in number of best results. However, the best outcomes of the experiment are achieved by vectors with dimension 300 in the benchmark MC-30.

From the point of view of the benchmarks, the similarities obtained with *wan2vec* achieved correlations of above 0.8 with the MC-30, RG-65, and MEN-TR-3k using both ϕ_{IF} and ϕ_{IAS} as weighting functions. This is a surprise given that MC-30 and RG-65 are the smallest sets, while MEN-TR-3k is the biggest one. In principle, we expected to have a better performance with the smaller testing sets. Probably, the features of the different test sets, including the year they were elaborated, the sociodemographic traits of the participants and the topic orientation of the words have an impact in the results.

On the other hand, the worst results are achieved with the SimLex-999 resource. This is not because of an small overlap – this is one of the best (96%). Given that the aim of the SimLex data set is to measure only similarity, the words psychologically related have the lowest scores. We believe that this issue affected the performance on this particular resource since in a WAN, words are supposed to be associated by relatedness. However, this idea is not consistent with the results we obtained with WS-353-SIM and WS-353-REL, where the former has outperformed the latter in every experiment.

In general, the higher scores are achieved with the MC-30. The fact that every word in this list is also in the EAT is not related to the result because it is also the best when the missing pairs are removed.

More research should be done about these results in the future in order to know the dynamics of the associations and to adjust the settings of the experiments.

An additional experiment has been conducted to test the performance and the adaptation of *wan2vec* to different similarity algorithms. We have reproduced the same tests that we have just described; but, this time, instead of measuring the correlation of two vectors by means of the well-known method of the cosine similarity, we have used an implementation of APSyn as it is introduced in [38]. This technique is defined as the extent of the weighted intersection between the top most salient contexts of the target words, weighting it by the average rank of the intersected features in the PPMI-sorted contexts lists of the target words. Therefore, the method is using a completely different algorithm to calculate the similarity between two vectors, and should provide a different perspective of the behavior of our node-based model.

Tables 3 and 4 show the results of APSyn with the same benchmarks that have been tested before, and with the same weighting functions ϕ_{IF} and ϕ_{IAS}. It is easy to see, when comparing them with 1 and 2, that the outcomes are very much lower with this method although, in general, ϕ_{IAS} achieves better results that ϕ_{IF}. Taking the averages, it can be seen that dimensions 50 and 200 work better with ϕ_{IF}, while the others achieve higher scores with ϕ_{IAS}. This is not exactly what happened with cosine similarity, that retrieved better results with 1000 dimensions with ϕ_{IF} and with 200 dimensions with ϕ_{IAS}.

Regarding the benchmarks, the best results are still the ones obtained with the list MC-30, but this time with ϕ_{IF} and dimension 50. Although for ϕ_{IAS} the best score is also achieved with this list, the needed dimension this time is 1000. In general, 50 and 200 are the best dimensions with ϕ_{IF} and 128 with ϕ_{IAS}. This last result would be consistent with the default setting of the node2vec resource.

4.2. Analysis of the node2vec's Walk Length

We performed an additional analysis which consisted in looking for the optimal walk length of the node2vec algorithm. The walk length indicates how deep the algorithm will walk through the graph in order to obtain the node's corresponding embedding. The default value is 80, and this is the one we used in the above experiments.

We measured the performance of the *wan2vec* embeddings carrying out the same experiments reported in the previous section, but systematically changing the length of the walk the node2vec algorithm traverses for finding the mapping for each node. We evaluated the walk length values from 20 to 200 in intervals of 10. Figure 2 presents the results of the experiments with two datasets the WS-353-REL (Relatedness) and the WS-353-SIM (Similarity).

In every case, the best results are achieved after the walk length of 60, reaching the best performance around 120. However, after this point, the improvement of the quality of the vectors is not remarkable and it can increase the time and complexity for training the model.

4.3. Comparison with Pretrained Vectors

In order to test and compare the quality of our *wan2vec* vectors, we also performed the experiments with pretrained vectors, which are presented in section 4.1. We selected three word embeddings models: word2vec, GloVe, and fastText. Table 5 presents the general features of the evaluated resources. In all cases the embeddings' dimension is 300 (dimensions), the difference of these three models basically lies in the type and size of the training corpus (Corpus (size)), the total amount of different words (Vocab. size), the training algorithm, and the context windows size. Data about the author of these models is also presented. The n(Overlap) column here is the same as in Tables 1-4,

[8]https://code.google.com/archive/p/word2vec/

[9]https://nlp.stanford.edu/projects/glove/

[10]https://nlp.stanford.edu/projects/glove/

[11]https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Benchmark	***n***	***n (Overlap)***	**Wan2vec Dimensions**						
			25	50	100	128	200	300	1000
MC-30	30	30	0.4455	**0.8053**	0.6364	0.6716	0.7287	0.5906	0.6268
WS-353-SIM	204	183	0.5382	0.5309	0.5322	0.6085	**0.6097**	0.5943	0.5323
MTurk-287	287	203	0.5335	0.5315	0.5111	**0.5653**	0.4805	0.5193	0.4667
WS-353-REL	253	214	0.3979	0.4467	0.4391	0.4795	**0.4965**	0.4271	0.4351
MEN-TR-3k	3000	2720	0.5507	**0.6370**	0.6153	0.6050	0.6149	0.5784	0.5517
SimLex-999	999	968	0.3049	0.3075	0.3506	0.3457	0.3617	0.3427	**0.3963**
MTurk-771	771	698	0.4921	0.5289	0.5354	0.5149	**0.5673**	0.5343	0.5331
RG-65	65	65	0.5332	**0.7609**	0.7341	0.6608	0.6496	0.5640	0.6207
Average			0.4433	0.5155	0.5156	0.5214	**0.5332**	0.4954	0.4942

Table 3

Spearman rank order correlation between wan2vec predictions (based on APSyn) and the evaluation benchmarks. The WAN graph was built using ϕ_{IF} as weighting function on the edges.

Benchmark	***n***	***n (Overlap)***	**Wan2vec Dimensions**						
			25	50	100	128	200	300	1000
MC-30	30	30	0.4608	0.6829	0.6075	0.6871	0.6816	0.6740	**0.7419**
WS-353-SIM	204	183	0.5802	0.5866	**0.6388**	0.6305	0.5580	0.6242	0.5745
MTurk-287	287	203	**0.6382**	0.5519	0.5629	0.5887	0.5243	0.5332	0.3547
WS-353-REL	253	214	0.4571	0.4432	**0.5164**	0.4874	0.4444	0.4992	0.4781
MEN-TR-3k	3000	2720	0.5909	0.6268	0.6344	**0.6351**	0.6017	0.6023	0.5691
SimLex-999	999	968	0.3013	0.3358	0.3696	0.3632	0.3682	0.3722	**0.4051**
MTurk-771	771	698	0.5244	0.4921	0.5251	**0.5593**	0.5246	0.5295	0.5526
RG-65	65	65	0.5121	0.5962	0.6713	**0.6857**	0.5741	0.5909	0.6338
Average			0.4714	0.4891	0.5229	**0.5355**	0.4888	0.5206	0.5170

Table 4

Spearman rank order correlation between wan2vec predictions (based on APsyn) and the evaluation benchmarks. The WAN was built using ϕ_{IAS} as weighting function on the edges.

tab:feqno1

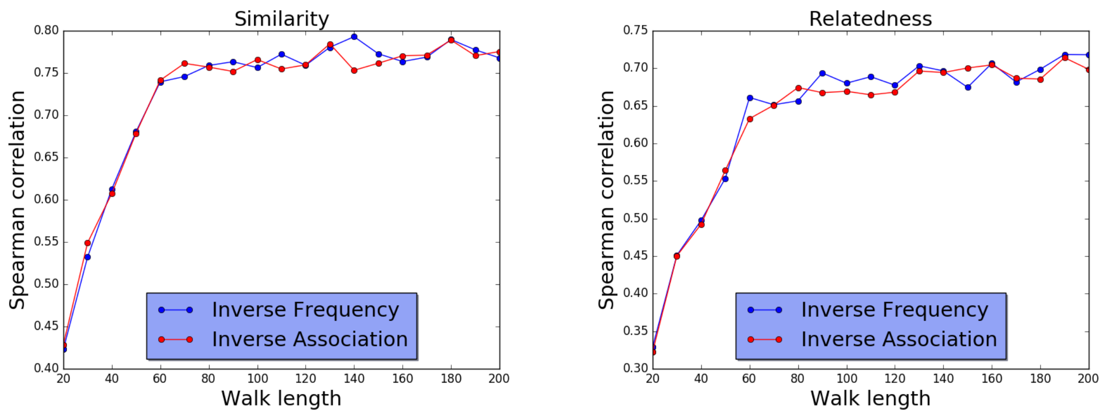

Fig. 2. Spearman rank order correlations obtained with different walk length using both graph weighting functions ϕ_{IF} and ϕ_{IAS}.

in this way we performed a fair evaluation using only the words that are available in all embedding models (including wan2vec).

Model file	**Dimen sions**	**Corpus (size)**	**Vocab. size**	**Author**	**Architec-ture**	**Training algorithm**	**Context window size**
Word2vec [8]	300	Google News (100B)	3M	Google	word2vec	negative sampling	BoW - 5
GloVe 6B[9]	300	Wikipedia + Gigaword 5 (6B)	400,000	GloVe	GloVe	AdaGrad	10+10
GloVe 42B[10]	300	Common Crawl (42B)	1.9M	GloVe	GloVe	GloVe	AdaGrad
FastText[11]	300	Wikipedia	Unknown	Facebook	FastText	Skip-gram	Character n-grams [1-5]

Table 5
Description of the pretrained vectors of the three evaluated word embeddings models.

Benchmark	*n*	*n(Overlap)*	**GloVe 6B**	**GloVe 42B**	**word2vec**	**fastText**	**wan2vec-300**	**wan2vec-100**
MC-30	30	30	0.7020	0.7770	0.78860	0.8119	**0.8582**	0.7912
WS-353-SIM	204	183	0.6564	0.6912	0.7644	0.7753	0.7623	**0.7770**
MTurk-287	287	203	0.7079	0.71851	0.7262	**0.7364**	0.6661	0.6976
WS-353-REL	253	214	0.6045	0.6212	0.6324	0.6886	0.6586	**0.7051**
MEN-TR-3k	3000	2720	0.7358	0.7338	0.7691	0.7649	0.7646	**0.8075**
SimLex-999	999	968	0.3703	0.3736	0.4413	0.3765	**0.5103**	0.4909
MTurk-771	771	698	0.6483	0.6808	0.6739	0.6719	0.6973	**0.7303**
RG-65	65	65	0.7695	0.8171	0.7607	**0.7998**	0.7913	0.7982
Average			0.6493	0.6766	0.6946	0.7032	0.7136	**0.7242**

Table 6
Cosine-based Spearman correlation between pretrained embeddings models, wan2vec with dimension 300 and wan2vec with its best correlation value. All of them regarding the benchmarks.

Table 6 shows the Spearman rank order correlations between the evaluated pretrained embeddings models and the human judgments (available in the benchmarks). The two last columns show the results obtained with *wan2vec*. One retrieves the outcomes of the method with vectors of dimension 300 with ϕ_{IAS} as weighting function. The last one has our best score, represented by the vectors with dimension 100 and ϕ_{IAS} as the weighting function. In average, the similarity obtained with the GloVe embeddings obtained the worst correlation scores, whereas the similarity obtained with fastText achieved of 0.7032 of Spearman correlation. These outcomes outperformed by *wan2vec* both with dimension 300 (0.7136) and 100 (0.7242).

The performance of fastText is better than *wan2vec* with dimension 300, with WS-353-SIM, MTurk-287, WS-353-REL, MEN-TR-3k and RG-65, whereas *wan2vec* with dimension 300 outperforms fastText with MC-30, SimLex-999 and MTurk-771. However, taking our best dimension, 100, *wan2vec* achieves better results than fastText except for MTurk-28 and RG-65. With these benchmarks, fastText gets always the best scores.

As for our *wan2vec* embeddings, the highest correlation was achieved with the MC-30 (300) and the MEN-TR-3k (100) datasets, obtaining correlations of 0.8582 and 0.8075, respectively. The pretrained vectors achieved the highest correlation of 0.8119 on the MC-30 with fastText embeddings and 0.8171 on the RG-65 dataset with GloVe 42B.

It is also interesting to note that the correlation with the SimLex-999 corpus is the lowest with all embeddings models, being the best the 0.51 of *wan2vec*-300 and the 0.49 of *wan2vec*-100, among the other methods.

Finally, the same tests have been performed with the APSyn similarity measure, with the results that are shown in Table 7. The columns follow the same structure than Table 6. In general, the results are very low, as it happened with *wan2vec* when using this measure (Tables 4 and 3). Besides this, the main difference between tables 6 and 7 is that, in the latter, GloVe 42B vectors are more competitive. Both GloVe 42B and word2vec achieve better results than fastText. *Wan2vec* with dimension 300 has consistent results, obtaining the best score with SimLex-999. And finally, if we take

Benchmark	*n*	*n(Overlap)*	**GloVe 6B**	**GloVe 42B**	**word2vec**	**fastText**	**wan2vec-300**	**wan2vec-128**
MC-30	30	30	0.6353	0.7272	0.7394	**0.7672**	0.6740	0.6871
WS-353-SIM	204	183	0.6241	**0.6880**	0.6829	0.6116	0.6242	0.6305
MTurk-287	287	203	0.5913	**0.6495**	0.5417	0.6276	0.5332	0.5887
WS-353-REL	253	214	0.4440	0.5933	**0.6029**	0.4530	0.4992	0.4874
MEN-TR-3k	3000	2720	0.5506	0.6054	0.6296	0.5873	0.6023	**0.6351**
SimLex-999	999	968	0.2233	0.3349	0.3680	0.2636	**0.3722**	0.3632
MTurk-771	771	698	0.5086	0.5819	**0.5879**	0.4887	0.5295	0.5593
RG-65	65	65	0.6485	0.6860	**0.7487**	0.6495	0.5909	0.6857
Average			0.5282	0.6083	**0.6126**	0.5561	0.5206	0.5355

Table 7

APSyn-based Spearman correlation between pretrained embeddings models, wan2vec with dimension 300 and wan2vec with its best correlation value. All of them regarding the evaluation benchmarks.

wan2vec with our best dimension, we obtain the highest scores with the benchmark MEN-TR-3k. However, GloVe 42B wins with WS-353-SIM and MTurk-287, whereas word2vec has the best results with WS-353-REL, MTurk-771 and RG-65. Finally, fastText achieve the best scores with MC-30.

Therefore, it is clear that the similarity measure between vectors we use has a clear impact on the results obtained. The models that are being compared show similar behaviors under similar tests, showing a decreasing and unexpected performance under APSyn measure. Indeed, we see an inversion of the scores obtained using cosine similarity. Wan2vec and fastText, that worked better with cosine, are the worst with APSyn, while GloVe and word2vec, the worst systems with cosine, obtain better numbers in APSyn.

5. Extrinsic evaluation

In order to evaluate the *wan2vec* embeddings in downstream tasks, we used SentEval[9] [51], a library for evaluating the quality of sentence embeddings. With this toolkit, we are able to assess the *wan2vec* embeddings generalization power by using them as features on several "transfer" tasks. In particular, we include in our evaluation:

- Sentiment analysis tasks (SST-2, SST-5): both binary and fine-grained SST [52].
- Paraphrase detection task (MRPC): aims at identifying if a pair of sentences captures a paraphrase/semantic equivalence relationship [53].
- Natural language inference task (SICK-E): consists in predicting whether two input sentences are entailed, neutral or contradictory [54].
- Semantic textual similarity tasks (STS'12-16): consists in evaluating how the cosine distance between two sentences correlate with a human-labeled similarity score through Pearson correlations.

For the classification tasks, the SentEval toolkit generates sentence vectors and a logistic regression classifier on top. For the unsupervised tasks (STS), it also generates sentence embeddings from the word embeddings.

Table 8 shows the evaluation of pretrained word vector models (GloVe and fastText) and our *wan2vec* vectors on several text classification tasks (SST, MRCP, and SICK-E). The results of *wan2vec* are around 10 points lower than the ones achieved by GloVe and fastText. In order to test if this is because the small coverage of vocabulary in our system, we have performed the evaluation with a lexical set for GloVe and fastText that was equivalent to *wan2vec*. As can be seen, the results are now comparable and, for tasks MRPC and SICK-E our system, with the weighting function ϕ_{IAS}, has obtained the best scores.

This experiment highlights that, although for the intrinsic evaluation the reduced vocabulary was not a problem, for the tasks contained in the SentEval toolkit this is a clear limitation.

Table 9 shows the evaluation of pretrained word vector models (GloVe and fastText) and our *wan2vec* vectors on the semantic textual similarity benchmarks (STS'12, STS'13, STS'14, STS'15, STS'16). The tests show the same general behaviour than the previous ones. With the entire vocabulary, our system is not competitive versus GloVe and fastText, while, with the

[9]https://github.com/facebookresearch/SentEval

Model	SST-2	SST-5	MRPC	SICK-E
GloVe 42B	79.01	45.23	72.62	79.01
FastText	**81.88**	**45.43**	**73.28**	**79.20**
GloVe (reduced)	73.31	37.19	70.55	76.72
FastText (reduced)	<u>74.30</u>	<u>38.46</u>	70.49	78.00
wan2vec ϕ_{IAS}	70.40	37.29	70.90	77.21
wan2vec ϕ_{IF}	70.35	38.19	<u>71.36</u>	<u>78.08</u>

Table 8

Transfer test results for pretrained embeddings models and *wan2vec* embeddings. All vectors have 300 dimensions.

same vocabulary, *wan2vec* reaches the best scores in every task except STS'13. It is worth to note that, following tables 8 and 9, the weighting function ϕ_{IF} seems to work better than ϕ_{IAS} for this task.

SentEval also includes a series of probing[10] [55] tasks to evaluate the linguistic properties encoded in a given sentence embedding. The tasks are divided in three groups, according to the type of information they encode:

– Surface information: evaluates the ability of the sentence embedding for preserving the surface properties of the original sentence. The sentence length (**SentLen**) task aims at predicting the length of sentences in terms of number of words. The word content (**WC**) task's goal is to predict which of the target words appear on the given sentence. In this way, it evaluates if it is possible to recover information about the original words in the sentence from its embedding.
– Syntactic information: evaluates the ability of sentence embeddings for preserving the syntactic properties of the sentences they encode. The bigram shift (**BShift**) task evaluate if the sentence embedding can distinguish the order of the words in the sentence. The tree depth (**TreeDepth**) task aims at predicting the maximum depth of a sentence's syntactic tree. The top constituent task (**TopConst**) tests the ability of the sentence embedding to capture latent syntactic structures by classifying the sequence of top constituents.
– Semantic information: the tasks in these group require to understand the meaning of a sentence. The **Tense** task aims at detecting the tense of the verb of the main clause (present or past). The subject number (**SubjNum**) task aims at predicting the number of the subject of the main clause (singular or plural). In the same way, the object number (**ObjNum**) task looks for the number of the direct object of the main clause. The semantic odd man out (SOMO) task evaluates whether a sentence occurs as-is in the source corpus, or whether a (single) randomly picked noun or verb was replaced with another form with the same part of speech. The coordination inversion (**CoordInv**) task aims at distinguishing between original sentence and sentences where the order of two coordinate clause conjunctions has been inverted.

The evaluation on the probing tasks are presented in Table 10. Again, the general tendency is that *wan2vec* only obtains comparable results when GloVe and fastText use the same amount of vocabulary; and, in the last case, fastText is always slightly better than *wan2vec*. Only for the task **TopConst**, our system gets better results with ϕ_{IF}. However, there is an exception with **SOMO**, that does not seem to be affected by the coverage of vocabulary. In this task, *wan2vec* with ϕ_{IF} outperforms the other systems. It is also remarkable how GloVe, when reduced, has very bad numbers with **SentLen**.

We want to note that we only tested *wan2vec* with dimension 300, which is not our best dimension. Probably using other dimensions that have obtained better results in intrinsic evaluation, such as 100, the behaviour of the model would improve. In spite of this, as a conclusion, the extrinsic evaluation has demonstrated that it is necessary to have large vocabularies to obtain competitive results with every type of task.

6. Conclusions and Future Work

In this paper, we introduce *wan2vec* a word embeddings model learned from Word Association Norms instead of large corpora. We applied the algorithm node2vec to a graph built over the Edinburgh Associative Thesaurus (EAT), a collection with 23,218 nodes. The result is a set of trained vectors that achieved better correlation with human judgments than other embedding models in the task of similarity and relatedness prediction.

The Node2vec algorithm learn embeddings using a random walk that explores the neighborhoods over the nodes to capture a better representation of the graph structure and the diversity of the connectivity.

As for the weight of the edges, we took into account two different functions: inverse frequency (ϕ_{IF}) and

[10]https://github.com/facebookresearch/SentEval/tree/master/data/probing

Model	**STS'12**	**STS'13**	**STS'14**	**STS'15**	**STS'16**
GloVe 42B	0.5208	0.4960	0.5460	0.5607	0.5144
FastText	**0.5826**	**0.5790**	**0.6491**	**0.6760**	**0.6427**
GloVe (Reduced)	0.3528	0.2430	0.4413	0.4639	0.3719
FastText (Reduced)	0.3468	0.3504	0.4773	0.4879	0.4001
Wan2vec ϕ_{IAS}	0.3411	0.3503	0.4858	0.5227	0.4127
Wan2vec ϕ_{IF}	0.3471	0.3499	0.4840	0.5291	0.4135

Table 9

Evaluation of sentence representations on the semantic textual similarity benchmarks. The average of Pearson correlations is used for STS'12 to STS'16 which are composed of several subtasks. All vectors have 300 dimensions.

Model	**SentLen**	**WC**	**TreeDepth**	**TopConst**	**BShift**	**Tense**	**SubjNum**	**ObjNum**	**SOMO**	**CoordInv**
GloVe 42B	**53.79**	**90.86**	31.63	63.16	**50.04**	**84.03**	78.28	76.65	49.74	**54.21**
Fasttext	50.48	92.05	**32.01**	**63.63**	49.74	86.59	**79.79**	**79.73**	49.74	53.17
GloVe (reduced)	27.16	70.72	25.25	32.75	49.61	70.52	72.09	73.50	49.11	51.85
Fasttext (reduced)	47.06	71.95	29.36	35.87	49.92	70.56	72.98	74.60	49.94	51.74
Wan2vec ϕ_{IAS}	43.78	69.51	29.18	36.38	49.26	68.66	70.12	71.74	49.77	51.46
Wan2vec ϕ_{IF}	44.63	69.13	29.33	37.13	49.49	68.05	69.81	71.95	**50.31**	50.97

Table 10

Probing task accuracies with Logistic Regression. All vectors have 300 dimensions.

inverse association strength (ϕ_{IAS}). The embeddings learned over the graph weighted with ϕ_{IAS} achieved a better average Spearman correlation than the ones learned in the graph with ϕ_{IF} weighting function.

We experimented with different walk length distances of node2vec, showing that the correlation tends to grow with higher distances and stabilizes above the length of 60. We believe that the default parameter of 80 is fair enough to have representative embeddings since higher values increase the complexity time of the experiments. Finally, we also evaluated a nodes pruning strategy, which consisted in keeping only those nodes that were strongly connected. Following the previous work by *Deyne et al.* [29] we kept only responses that also occur as stimuli and stimuli that were also given as responses. It resulted in a high loss of outcoming nodes and a high reduced overlap, that did not let us give a strong enough comparative benchmark.

The results we are reporting clearly outperform the ones obtained with some well-know pretrained vectors (word2vec, GloVe, FasText) trained in large corpora, like *Wikipedia*. This results are in line with the work by *Deyne et al.* [29], reaffirming the importance of the Word Association Norms as a resource for natural language processing tasks. This kind of resources reflect, to some extent, the mental representation of the words. Additionally, they are valuable for learning distributional representation of words.

On the contrary, the worst performance of the model has been reported when dealing with extrinsic evaluation. The tests we have conducted lead us to think that the reduced vocabulary is the main reason of these results. Therefore, this is one of the main issues the model has to deal with, as we explain in what follows.

One of the most important limitations of the model is its dependency of WAN compilations. There are two important issues related to that. First, collecting WANs is a hard and time consuming task. Not every language has this type of resources and the capability to collect it. Anyway, other methods based in corpus can take advantage of large digital collections of texts, like Wikipedia, online newspapers, etc., while WANs need an experimental approach, planning and very a specific treatment. Second, the number of words in WANs is very small compared with other corpora. As we have explained, in the larger resources based in associations, like EAT, the overlap with the benchmarks is high, but smaller than the obtained in methods that use large corpora. When comparing the performance of *wan2vec* with the others, it seems clear that the total number of tokens is not necessarily an advantage. Moreover, corpora like Wikipedia have many noisy elements that do not improve the behaviour of the vectors. However, it would be optimal to have models that could include a larger set of different words.

A possible solution for this problem would be to automatically generate word association norms between pairs retrieved by a medium-size corpus (i.e., BNC), and build a new resource that can account for syntactic, semantic and cognitive connections between words. There are some contributions [28] that have introduced interesting ideas on how a system can learn word associations from a WAN and extend it to larger collections. This is a task that we plan to develop in the future.

References

[1] S.T. Dumais, Latent Semantic Analysis, *Annual Review of Information Science and Technology* **38**(1) (2004), 188–230.

[2] T. Mikolov, K. Chen, G. Corrado and J. Dean, Efficient Estimation of Word Representations in Vector Space, *Computing Research Repository* **arXiv:1301.3781** (2013). https://arxiv.org/abs/1301.3781.

[3] Z.S. Harris, Distributional Structure, *Word* **10**(2–3) (1954), 146–162.

[4] J. Pennington, R. Socher and C.D. Manning, GloVe: Global Vectors for Word Representation, in: *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. http://www.aclweb.org/anthology/D14-1162.

[5] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching Word Vectors with Subword Information, *Computing Research Repository* **arXiv:1607.04606** (2016). https://arxiv.org/abs/1607.04606.

[6] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch and A. Joulin, Advances in Pre-Training Distributed Word Representations, in: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, LREC'18, 2018.

[7] C.K. Ogden, *Basic English: A General Introduction with Rules and Grammar*, Paul Treber. http://ogden.basic-english.org/be0.html.

[8] M. West, *A General Service List of English Words*, Longman, 1953.

[9] C. Browne, B. Culligan and J. Phillips, New General Service List (2013). http://www.newgeneralservicelist.org/.

[10] D.A. Kwary and Jurianto, Selecting and Creating a Word List for English Language Teaching, *Teaching English with Technology* **17**(1) (2017), 60–72.

[11] R. Flesch, How Basic Is Basic English?, *Harper's Magazine* **188:1126**, 339–343.

[12] Longman (ed.), *Longman Dictionary of Contemporary English*, Longman, 2003.

[13] G.A. Miller and W.G. Charlees, Contextual Correlates of Semantic Similarity, *Language and cognitive processes* **6**(1) (1991), 1–28.

[14] S. De Deyne, D.J. Navarro and G. Storms, Associative Strength and Semantic Activation in the Mental Lexicon: Evidence from Continued Word Associations, in: *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, Cognitive Science Society, 2013.

[15] J. Borge-Holthoefer and A. Arenas, Navigating Word Association norms to Extract Semantic Information, in: *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, 2009.

[16] A. Grover and J. Leskovec, node2vec: Scalable Feature Learning for Networks, in: *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 855–864.

[17] A.M. Collins and E.F. Loftus, A spreading-activation theory of semantic processing., *Psychological review* **82**(6) (1975), 407.

[18] C. H.H., Word Associations and Linguistic Theory, in: *New Horizons in Liguistics*, John Lyons, 1975.

[19] J.F. Sowa, Conceptual graphs as a universal knowledge representation, *Computers & Mathematics with Applications* **23**(2) (1992), 75–93.

[20] J. Aitchison, *Words in the mind: An introduction to the mental lexicon*, John Wiley & Sons, 2012.

[21] G.H. Kent and A.J. Rosanoff, A Study of Association in Insanity, *American Journal of Insanity* **1910**(67) (1910), 317–390.

[22] I. Istifci, Playing with Words: a Study of Word Association Responses, *Journal of International Social Research* **3**(10) (2010).

[23] M. Lafourcade, Making people play for Lexical Acquisition with the JeuxDeMots prototype, in: *SNLP'07: 7th International Symposium on Natural Language Processing*, 2007, pp. 13–15.

[24] G.R. Kiss, C. Armstrong, R. Milroy and J. Piper, *An Associative Thesaurus of English and its Computer Analysis*, Edinburgh University Press, Edinburgh, 1973.

[25] D.L. Nelson, C.L. McEvoy and T.A. Schreiber, *Word Association Rhyme and Word Fragment Norms*, The University of South Florida, 1998.

[26] G. Bel-Enguix, R. Rapp and M. Zock, A Graph-Based Approach for Computing Free Word Associations., in: *Proceedings of the 9th edition of the Language Resources and Evaluation Conference*, LREC'14, 2014, pp. 221–230.

[27] A. Garimella, C. Banea and R. Mihalcea, Demographic-aware word Associations, in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP'17, 2017, pp. 2285–2295.

[28] A. Sinopalnikova and P. Smrz, Word Association Thesaurus as a Resource for extending Semantic Networks., in: *Communications in Computing*, 2004, pp. 267–273.

[29] S. De Deyne, A. Perfors and D.J. Navarro, Predicting Human Similarity Judgments with Distributional Models: The value of word associations, in: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1861–1870.

[30] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Pasca and A. Soroa, Proceedings of NAACL-HLT 2009, in: *A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches*, 2009.

[31] T. Pedersen, S. Patwardhan and J. Michelizzi, WordNet::Similarity: Measuring the Relatedness of Concepts, in: *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, Association for Computational Linguistics, Stroudsburg, PA, USA, 2004, pp. 38–41. http://dl.acm.org/citation.cfm?id=1614025.1614037.

[32] M.A. Hadj Taieb, M. Ben Aouicha and A. Ben Hamadou, A New Semantic Relatedness Measurement Using WordNet Features, *Knowl. Inf. Syst.* **41**(2) (2014), 467–497, ISSN

0219-1377. doi:10.1007/s10115-013-0672-4. http://dx.doi.org/10.1007/s10115-013-0672-4.

[33] M. Faruqui, J. Dodge, S. Jauhar, C. Dyer, E. Hovy and N. Smith, Retrofitting word vectors to semantic lexicons, in: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2015, pp. 1606–1615.

[34] Y. Cai, Q. Zhang, W. Lu and X. Che, A hybrid approach for measuring semantic similarity based on IC-weighted path distance in WordNet, *Journal of Intelligent Information Systems* **51**(1) (2018), 23–47. doi:10.1007/s10844-017-0479-y. https://doi.org/10.1007/s10844-017-0479-y.

[35] M. Ben Aouicha, M.A. Hadj Taieb and A. Ben Hamadou, SISR: System for integrating semantic relatedness and similarity measures, *Soft Computing* **22**(6) (2018), 1855–1879, ISSN 1433-7479. doi:10.1007/s00500-016-2438-x. https://doi.org/10.1007/s00500-016-2438-x.

[36] E. Gabrilovich and S. Markovitch, Computing semantic relatedness using Wikipedia-based explicit semantic analysis, in: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, 2007, pp. 1606–1611.

[37] M. Hadj Taieb, M. Ben Aouicha and A. Ben Hamadou, Computing semantic relatedness using Wikipedia features, *Knowl Based Syst* **50** (2013), 260–278.

[38] E. Santus, E. Chersoni, A. Lenci, C.-R. Huang and P. Blache, Testing APSyn against Vector Cosine on Similarity Estimation, in: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,Osaka, Japan, December 11-17 2016*, 2016, pp. 1861–1870.

[39] E. Santus, H. Wang, E. Chersoni and Y. Zhang, A Rank-Based Similarity Metric for Word Embeddings, in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), Melbourne, Australia, July 15 - 20, 2018*, Association for Computational Linguistics, 2018, pp. 552–557.

[40] D.L. Nelson, V.M. McKinney, N. Gee and G. Janczura, Interpreting the Influence of Implicitly Activated Memories on Recall and Recognition, *Psychological Review* **105**(2) (1998), 299–324.

[41] T. Schnabel, I. Labutov, D. Mimno and T. Joachims, Evaluation Methods for Unsupervised Word Embeddings, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP'15, 2015, pp. 298–307.

[42] H. Gómez-Adorno, I. Markov, G. Sidorov, J. Posadas-Durán, M.A. Sanchez-Perez and L. Chanona-Hernandez, Improving Feature Representation based on a Neural Network for Author Profiling in Social Media Texts, *Computational Intelligence and Neuroscience* **2016** (2016), 13.

[43] H. Gómez-Adorno, J.-P. Posadas-Durán, G. Sidorov and D. Pinto, Document Embeddings Learned on Various Types of n-grams for Cross-topic Authorship Attribution, *Computing* (2018), 1–16.

[44] M. Baroni, G. Dinu and G. Kruszewski, Don't count, predict! A Systematic Comparison of context-counting vs. context-predicting Semantic Vectors, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1, 2014, pp. 238–247. http://www.aclweb.org/anthology/P14-1023.

[45] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman and E. Ruppin, Placing Search in Context: The Concept Revisited, in: *Proceedings of the 10th International Conference on World Wide Web*, WWW'01, ACM, 2001, pp. 406–414.

[46] F. Hill, R. Reichart and A. Korhonen, Simlex-999: Evaluating semantic models with (genuine) similarity estimation, *Computational Linguistics* **41**, 665–695.

[47] K. Radinsky, E. Agichtein, E. Gabrilovich and S. Markovitch, A word at a time: computing word relatedness using temporal semantic analysis, in: *Proceedings of the 20th international conference on World wide web*, ACM, 2011, pp. 337–346.

[48] E. Bruni, G. Boleda, M. Baroni and N.-K. Tran, Distributional semantics in technicolor, in: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, Association for Computational Linguistics, 2012, pp. 136–145.

[49] G. Halawi, G. Dror, E. Gabrilovich and Y. Koren, Large-scale learning of word relatedness with constraints, in: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2012, pp. 1406–1414.

[50] H. Rubenstein and J.B. Goodenough, Contextual correlates of synonymy, *Communications of the ACM* **8** (1965), 627–633.

[51] A. Conneau and D. Kiela, SentEval: An Evaluation Toolkit for Universal Sentence Representations, *arXiv preprint arXiv:1803.05449* (2018).

[52] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng and C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[53] B. Dolan, C. Quirk and C. Brockett, Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources, in: *Proceedings of the 20th international conference on Computational Linguistics*, Association for Computational Linguistics, 2004, p. 350.

[54] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, R. Zamparelli et al., A SICK cure for the evaluation of compositional distributional semantic models., in: *LREC*, 2014, pp. 216–223.

[55] A. Conneau, G. Kruszewski, G. Lample, L. Barrault and M. Baroni, What you can cram into a single vector: Probing sentence embeddings for linguistic properties, *arXiv preprint arXiv:1805.01070* (2018).