Distributional memory explainable word embeddings in continuous space

Lauro Snidaro, Giovanni Ferrin and Gian Luca Foresti
Department of Mathematics, Computer Science and Physics
University of Udine
Via delle Scienze, 206, 33100 Udine, Italy
{lauro.snidaro, giovanni.ferrin gianluca.foresti}@uniud.it

***Abstract*—Natural Language Processing (NLP) is a key processing step in fusion systems that need to process unstructured -and possibly human generated- text in natural language. Recent developments in Deep Learning have greatly increased the performance of NLP tasks. In particular, learned word representations have the form of high dimensional real valued vectors, called *word embeddings*, that have a number of amenable properties such as representing similar words with similar values of their vectorial representation, and capturing semantic regularities that correspond to geometric properties in the continuous high dimensional space. However, word embeddings have the drawback of being non interpretable. That is, their dimensions cannot be clearly associated to linguistic features. In this work, we propose real valued explicit linguistic word vectors that enjoy the properties of learned word embeddings while being human understandable.**

***Index Terms*—Word Embeddings, NLP, Explainable Artificial Intelligence, Interpretability, Situation assessment, Semantics, Lexical semantics, Distributional semantics**

I. Introduction

The present paper represents a further step in outlining the possibility of exploiting natural language text corpora. The aim consists in bringing out similarity mappings able to drive the retrieval of semantic information about real world, both about objects and events [1]. Recent examples in the literature can be found that discuss, for example, automatic identification of artifact usage information [2] from corpora.

Trying to get meaning from text implies dealing with the problem of defining what "meaning" is with respect to lexical items. Historically, the point has been addressed talking about *Extensional meaning* (all the things in the world that make a term or predicate true) and *Intensional meaning* (the conditions under which a predicate is true), however not solving the problem of how to define the "conditions". More recent research in the field is grounded on the idea that to perform lexical semantics, a scientifically correct theory of the world is not needed, but people's commonsense theories.

Arguing about the distinction between lexical knowledge and world knowledge is far beyond the scope of our research. We agree with Hobbs [3]: "If the brain makes a distinction between linguistic and world knowledge, it does not appear to be reflected in the temporal course of processing language".

II. Modeling textual knowledge

The model of semantic representation we are looking at to perform the proposed operations, is the Distributional approach, based on the assumption that lexical meaning depends on the contexts in which lexemes are used, and grounded on

- the philosophical "usage-based" view of meaning ("The meaning of a word is its use in the language") developed by Ludwig Wittgenstein in his later writings [4]
- the idea of habitual word combinations: "You shall know a word by the company it keeps!" of John R. Firth [5]
- the Distributional hypothesis: difference of meaning correlates with difference of word distribution in text [6]

Thus Distributional semantics is based on the assumption that the statistical distribution of linguistic items in context characterizes their semantic behavior. A summarization of its main characters can be found in [7].

The availability of large natural language corpora and the refinement in machine learning algorithms has increased the interest in working with such semantic representations, not only on typically linguistic tasks, but also in trying to predict complex and practically relevant human evaluations in domains as social cognition, health behavior, risk perception, and organizational behavior [8].

A. Distributional representations

A distributional representation consists in a matrix built with the following typical procedure (a thorough review and state of the art in distributional semantics can be found in [9]):

- extract and count, from a corpus, co-occurrences between target lexical items and linguistic contexts (collocates);
- represent the distribution of lexical items in the form of a matrix, whose row vectors correspond to target lexical items, columns to contexts, and the entries to their co-occurrence frequency;
- transform frequencies into weights, significative of the importance of the contexts;
- measure the semantic similarity between lexemes evaluating the similarity between the row vectors in the co-occurrence matrix.

Building distributional representations requires some parameters to be determined:

- the selection of target lexemes;
- the definition of context type;
- the choice of weighting scheme;
- the (possible) application of some dimensionality reduction;
- the choice of a vector similarity metric;

and, in particular, a distributional semantic model (DSM) requires a peculiar configuration of the mentioned parameters. Among the possible models, the main differences can be found along the dimensions of type of context and method of learning distributional vectors.

The choice of context strongly affects the similarity relations that can be identified. The *word models* and *region models* represent the most followed approaches to determine semantic similarity. In word models two lexemes are considered to be similar if the collocates they co-occur with are statistically the same. In region models, instead, two lexemes are similar if they tend to co-occur in the same texts. Word models, on one hand, focus on terms that share a number of common attributes (like `coast` and `shore`), that is *attributionally similar*. Region models, on the other hand, focus on terms that appear in texts about the same arguments, belonging to the same domain (like `clothes` and `closet`), that is *topically similar*.

As to methods to learn the distributional representations, the choice is between *count models* and *prediction models*. The former, more precisely the *matrix models* subtype, represent target lexemes by counting their co-occurrences in linguistic contexts. The data are arranged into co-occurrence matrices that formally represent the distributional statistics extracted from the corpus. Some models directly implement the basic extraction to build the representations (*window models*), some other introduce extensions and variants to the method, for example taking into consideration lexico-syntactic patterns (*syntactic models*).

B. Explicit vs implicit representations

In the representations obtained, therefore, each vector dimension represents the count of the co-occurrence of lexemes with the specific context. These vectors can reach very high dimensions, as numerosity of contexts in language data is usually very high, and, due to Zipf law (the frequency of words occurrences in a corpus is inversely proportional to words rank [10]), tend to be very sparse.

High-dimensionality have the consequence of leading to miss generalizations in data. As each context represent a distinct feature, explicit representations do not take into account mutual similarity and correlation of contexts. Possible co-occurrences, moreover, could remain unobserved in corpora because of the uneven data distribution. This kind of representation, obtained without any further processing, is defined as an *explicit* distributional vector.

A possible solution to the aforementioned problems consists in representing lexemes with distributional vectors which differ from the former by having a much smaller dimensionality (few hundreds), corresponding to latent features extracted from co-occurrences, and being dense, as most of the values are nonzero. In these vectors there is no direct correspondence between features and linguistic contexts and they are referred to as *implicit* distributional vectors.

Implicit distributional representations are commonly obtained mapping the co-occurrence matrix onto a reduced latent semantic space using matrix reduction algorithms.

Low-dimensional implicit representations can also be obtained without counting co-occurrences but just using neural network algorithms. These representations, referred to as *(neural) word embeddings*, are vectors whose weights are set to optimally predict the contexts in which the corresponding words tend to appear. Since similar words occur in similar contexts, the system naturally learns to assign similar vectors to similar words [11], such models are known as *prediction models*.

C. AI Explainability

Explainability of components is a key difference between explicit and implicit representations: explicit vectors, whose dimensions show a direct correspondence between features and linguistic contexts, can be interpreted directly. It is difficult (if not impossible), on the other hand, to assign semantic values to latent features.

Interpretability makes explanation possible. The entire history of Western philosophy shows that explanation is strictly connected to causal reasoning which in turn plays a central role in using knowledge and beliefs to build mental models about states of things and events. Mental models might even be defined as the causal network we believe as ruling external world and making things happen [12], we need interpretability to understand the world and to decide and act upon it.

The last decade research in machine learning has shown a rapid evolution in techniques like neural networks. The increase in sophistication has lead to a lack of interpretability of algorithms but, at the same time, the market and the policy makers (see the European Union General Data Protection Regulation, Article 22 [13]) demand explainability of the resulting decisions. Recent studies (e.g. [14]) address the urgency of interpretability and explainability for a user, in understanding the outcome of machine learning algorithms, and research is also ongoing into interpretability of word embeddings (see [15]).

Each of the aforementioned models have some advantages and suffer from drawbacks, and no particular approach seems to definitely outperform in semantic tasks like similarity judgments, synonym detection or noun categorization (see again [9] and [16], [17], [18], [19], [20]). On these premises, among the different models proposed within the Distributional approach we have decided to focus our attention on the Distributional Memory framework [21], a syntactic, count, matrix model which extracts distributional information from the corpus in the form of a set of weighted *word-link-word* ternary tuples arranged into a third-order tensor [22].

TABLE I

EXAMPLES OF DM TUPLES FOR THE NOUN 'KNIFE' LINKED TO THE VERBS 'CUT', 'USE', AND 'WATCH'.

Word1	**Link**	**Word2**	**Value**
knife-n	with	cut-v	1098.8484
knife-n	obj	use-v	1061.7598
knife-n	obj	watch-v	0.5139

TABLE II

EXAMPLES OF DM TUPLES FOR THE NOUN 'KNIFE' LINKED TO THE ADJECTIVES 'SHARP' AND 'BLUNT'

Word1	**Link**	**Word2**	**Value**
knife-n	nmod-1	sharp-j	156.6379
knife-n	is	sharp-j	80.0795
knife-n	nmod-1	blunt-j	69.6651

III. THE DM TENSOR

The DM framework represents the distributional information in a corpus as a third order tensor, that is a ternary geometrical object that takes the form of weighted word-link-word tuples encoding typed co-occurence relations among words [22]. Formally, let W be a set of strings representing words, and L a set of strings representing co-occurrence links between words in a text, then $T \subseteq W \times L \times W$ is the set of tuples $t = \langle w_1, l, w_2 \rangle$ indicating that w_1 co-occurs with w_2 in the corpus via the relation encoded by l. Each tuple t is associated to a value v (also called weight) which is dependent on the number of co-occurences in the corpus of w_1 and w_2 via l.

For example, the DM tensor contains the tuple $\langle \texttt{knife-n}, \texttt{sbj\_tr}, \texttt{cut-v}, 171.8297 \rangle$ that indicates that the word `knife` (the suffix -n encodes the fact that the word is a noun) occurs in the corpus as subject of the transitive verb (link `sbj_tr`) `cut` (the -v suffix encodes the fact that the word is a verb) with weight 171.8297.

The reader is referred to [22] for details on how the value is computed, for the discussion here it is sufficient to say that the greater the weight the greater the number of co-occurrences in the corpus of the two words indicated by the tuple via the link. In the DM tensor, 2678 tuples can be found associating the example noun with a verb via a given link. Other examples can be seen in Table I, where the first row tells us that in the corpus the act of cutting is often performed by means of a knife, or that with the same frequency we can find that `knife` is the direct object of the verb `use`. The third row indicates that very seldom `knife` occurs as the direct object of the verb `watch`.

The lower bound for the weight is zero, theoretically indicating the absence of co-occurences for a given tuple. No such tuples are present in the DM tensor, that accounts only for positive co-occurences in the corpus.

The tensor contains ca. 130M tuples, the number of nouns is 20408 and the number of different links is 25336. It is interesting to highlight, for the reminder of the paper, that nouns are linked to adjectives as well. For example, in Table II a few tuples are shown involving again the noun `knife`, but linked this time to the adjectives `sharp` and `blunt`. There are 917 tuples in the tensor indicating co-occurences of `knife` with adjectives. Those shown in Table II are among the ones with greater weight value. The first row of the table indicates that the example noun is frequently associated to the adjective `sharp` (the suffix -j indicates that the word is an adjective) via the (inverse) relation `nmod` (noun modifier). The second row reports that `knife` is also frequently associated to `sharp` via the link `is`, although the frequency is almost half with respect to the previous tuple. The last tuple of the table shows that with even less frequency `blunt` is a noun modifier for `knife` in the corpus.

What it is interesting to observe here is that the adjectives are used in the language to describe *properties*. This will be exploited in the approach proposed in Sec. V.

IV. WORD EMBEDDINGS

The recent upsurge of deep learning techniques has driven the development of new models for NLP. In particular, the representation of words as a numerical vectors has been around for more than forty years, however only recently [23] the idea of learning, via unsupervised algorithms, n-dimensional vectors that would capture word-meaning and context in their values was developed. Therefore, a set of word vectors for a vocabulary is able to capture the meaning of words, the relationship between them and their context.

Vector representations embed words in a feature space, thus the name *word embeddings*. As points in a n-dimensional space, word embeddings enjoy a number of interesting properties such as:

- a unique representation (vector) for each word
- vectors are relatively low-dimensional, with typically only a few hundred dimensions
- words with similar meaning have similar vector values and are thus close-by in the n-dimensional feature space
- semantic regularities correspond to geometric properties

Despite all the advantages described above, learned word embeddings have the drawback of being non interpretable. That is, their dimensions cannot be clearly associated to linguistic features or properties.

V. LINGUISTIC WORD EMBEDDINGS

The proposed approach is to build *linguistic word embeddings*, that is vectorial representations (embeddings) of the words of a language. In our case here, we will limit ourselves to the representation of English nouns. As already mentioned, the novelty of the proposed representation is to be completely interpretable as every component (dimension) of the embedding vector corresponds to an adjective in the English language. Let n_A be the number of adjectives in the language we want to model, then the dimensionality of our real-valued vectors (word embeddings) will be n_A and their components will be associated to the adjectives in the language in alphabetical order. Therefore, the j-th word in the vocabulary is represented by the j-th column vector in the embedding matrix as follows:

$$e_j = \begin{bmatrix} adj_{1,j} \\ adj_{2,j} \\ \vdots \\ adj_{n_A,j} \end{bmatrix} \tag{1}$$

$adj_{i,j}$ is then i-th component of e_j and stores the value of the i-th adjective for word j. For example, `knife-n` is the 10343-th entry in the $Nouns$ vocabulary and a partial representation of its embedding is the following, where the actual computed values are shown corresponding to the the adjectives on the right:

$$e_{10343} = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} 0 \\ 4.329 \\ 16.368 \\ \vdots \\ 73.297 \\ \vdots \\ 327.629 \\ \vdots \\ 0 \end{bmatrix} \begin{array}{c} \texttt{Aboriginal-j} \\ \texttt{Afghan-j} \\ \texttt{African-j} \\ \vdots \\ \texttt{long-j} \\ \vdots \\ \texttt{sharp-j} \\ \vdots \\ \texttt{zealous-j} \end{array}$$

The computation of the embedding values and the process for building vectorial representation of our proposed Linguistic Word Embeddings are described in the following section.

A. Computing the linguistic word embeddings

Building our linguistic word embeddings is a one-off procedure performed on the DM tensor in three steps:

1) Extraction of the list of nouns
2) Extraction of the list of adjectives
3) Construction of the embedding matrix

To simplify accessing and querying the DM tensor, we saved in SQL database format the DM text file[1]. We extracted from the DM tensor a vocabulary of 20408 nouns and a vocabulary of 5253 adjectives. This could be easily done by running SQL queries on the DM database selecting all distinct entries of $Word1$ (or $Word2$ alternatively) ending with suffix '-n' and '-j' respectively. The resulting single column dataframes $Nouns$ and $Adjectives$ are then saved as tables to be used in the following step.

To carry out the third step, we developed a parametric SQL query to be run via Python for each noun in $Nouns$. The query looks for all tuples involving the noun at hand linked to an adjective and saves the results in a temporary table, specifically a Common Table Expression (CTE), called $NounAdjCTS$. The CTS has tuples of the form $\langle adjective, value \rangle$. However, the adjectives found in this table are only a subset of all the entries in the $Adjectives$ vocabulary. Since, as explained in Section V, we need a fixed-dimension vectorial representation with all the adjectives, a LEFT JOIN[2] is performed between the $NounAdjCTS$ and $Adjectives$ tables thus resulting in a table with the same number of entries as $Adjectives$. This single column table is then cast into a column vector of floats and added as the i-th column of the embedding matrix, where i is the index of the noun in the $Nouns$ vocabulary.

Listing 1 reports an instance of the parametric query developed for computing the linguistic embeddings for all the nouns in our vocabulary. This particular instance of the query returns the values (weights) for each adjective as found in the DM tensor for `knife-n`. It is worth remembering that in the DM tensor a certain noun can be found linked to the same adjective via different links, as exemplified in Table II where `knife-n` reported to be linked to `sharp-j` via `nmod-1` and via `is` with different weights. What value should then be used in the embedding vector for `knife-n` in the component corresponding to `sharp-j` ? We experimented with different functions as taking the maximum value or taking the sum of the values, however we didn't notice significant differences in the performance of the embeddings in the experiments. The sum function, used in Listing 1, obviously returns higher values than the max function and, as observed in the experiments we performed, this translated in higher similarity ratings between couple of vectors. This is a good thing when the two words are actually similar, but this effect is balanced by the drawback of higher ratings also for couples of words that should not be considered similar.

The embedding matrix created with the procedure described above is very sparse. For example, the matrix generated with instances of the SQL query in Listing 1 for each noun in the vocabulary as $Word1$ is 93.58% sparse, with a mean number of non-zero components per column (embedding vector) of 336.98. Our 5253 embedding dimension seems therefore largely unused and in contrast to the principle behind learned word embeddings that aim to dimensionality reduction (with respect for example to one-hot vectors) as seen in Sec. IV. However, it should be noticed here that, in this work, our embeddings are not learned but computed directly from the DM tensor. As already said, the computation is a one-off procedure that takes a few minutes on a modern laptop machine, and the matrix is an 817MB structure when saved to disk in .npy format. As a reference, the DM tensor is instead a 4GB file. In addition, tremendous savings can be obtained when representing the embedding matrix via sparse structures, as for example those made available by scipy.sparse in Python.

VI. EXPERIMENTS

In this section, we experiment with a typical task for measuring the performance of word embeddings: word similarity. In particular, we perform a few different word similarity tasks:

1) looking for the most similar entries in the dataset to a given word;

[1]http://clic.cimec.unitn.it/dm/

[2]The LEFT JOIN operation returns all records from the left table and the matched records from the right table, possibly with NULL results from the right side when no match is detected.

```sql
WITH NounAdjCTS (Adj,Value) AS
(
        SELECT Word2 as Adj, SUM(Value) as Value
        FROM dm_table
        WHERE Word1='knife-n' and Word2 LIKE '%%-j'
        GROUP BY Word2
)
SELECT NounAdjCTS.Value
FROM Adjectives LEFT JOIN NounAdjCTS ON NounAdjCTS.Adj = Adjectives.Adj
```

Listing 1. Instance of the parametric SQL query developed for extracting the linguistic embeddings from the DM tensor. In this case, the query returns the embedding of knife-n. The first part of the query is comprised by a CTE named NounAdjCTS that extracts all the tuples where the noun parameter is linked to an adjective. The main part of the query performs a left join between the Adjectives table and NounAdjCTS.

2) measuring the correlation between the similarities computed by the embeddings on a set of word couples from a dataset and the average results obtained by humans on the same words.

In addition to these experiments, we would like also to reflect on the explainability of the obtained results which is the driving feature of the proposed approach.

A. Most similar words

For this task we have selected a set of nouns representing both simple and complex objects as well as abstract concepts. Fig. 1 shows the most similar embeddings to the nouns `knife, ramp, house, kid, computer, faith` via the cosine similarity measure. The cosine similarity between two vectors u and v is defined as:

$$cos(u,v) = \frac{u \cdot v}{\|u\| \, \|v\|} = \frac{\sum_i u_i v_i}{\sqrt{\sum_i {u_i}^2}\sqrt{\sum_i {v_i}^2}} \tag{2}$$

The cosine similarity is widely used when working with embeddings. It has a few interesting properties that make it often preferable over other norms. While in principle returning values in the $[-1, 1]$, the function is strictly bounded in $[0, 1]$ when the vectors are positive as in our case. The measure is also insensitive to scale and works well with sparse vectors.

Therefore, the similarity between the embedding of each noun in the mentioned list and all the other embeddings has been computed and the top seven results are shown in the figure. It can be noticed that the proposed embedding representation is able to capture the characteristics of each noun making it comparable with all the others via distance measures. This is a typical capability of learned word embeddings, however we would like to stress again that the approach here is very different and hinges on the linguistic features (specifically adjectives) that are similar to those that a human would use to describe these objects.

We can see that the most similar words in the database denote a referent in the real world that intuitively shares physical qualities with the query noun. While in classical learned word embeddings the intuition would remain such, we will see in Sec. VI-C examples of the main characteristics that determine the similarity between couples of vectors. For the moment, it is interesting to notice that even a noun related to a concept such as `faith-n` is properly handled and considered close to `belief-n`.

In all examples, it can be seen that while the closest results are effectively similar to the query noun, results further away are linked to the query more due to some degree of relatedness rather than actual similarity. This happens in learned word embeddings as well.

B. Correlation with similarity DataSets

We perform the evaluation of our approach on four of the publicly available, state of the art, Test collections. All of them have been built following the "contextual hypothesis" with the purpose of measuring *attributional similarity* (the degree of synonymy between two words) and association, or *relatedness*, (the degree to which words are associated via any kind of semantic relationship, including synonymy, meronymy, hyponymy, hypernymy, functional or associative).

- RG-65 Test Collection [24]. It contains 65 pairs of nouns together with a value from 0.0 to 4.0 assigned by judges after a process of ordering, according to the amount of "similarity of meaning".
- MC-28 Test collection [25]. It is a subset of 30 noun pairs from the RG-65 set, 10 of which selected from the high level (rated 3 to 4), 10 from the intermediate level (rated 1 to 3), and 10 from the low level (rated 0 to 1) and rescored.
- WordSimilarity-353 (WS-353) Test Collection [26]. It contains two sets of 353 English noun pairs representing various degrees of relatedness, along with human-assigned similarity judgements. The present dataset includes all the noun pairs of the MC-28, with diffrent scores on a scale from 0 (semantically unrelated words) to 10 (identical words)
- SimLex-999 (SL-999) Test collection [27]. It contains 999 pairs of words. The dataset, differently from the previuos, explicitly quantifies similarity rather than association or relatedness and contains a range of adjective, verb and noun concept pairs covering the full concreteness spectrum: words denoting entities that are associated but not actually similar have a low rating. For example the pair of similiar words `coast - shore`

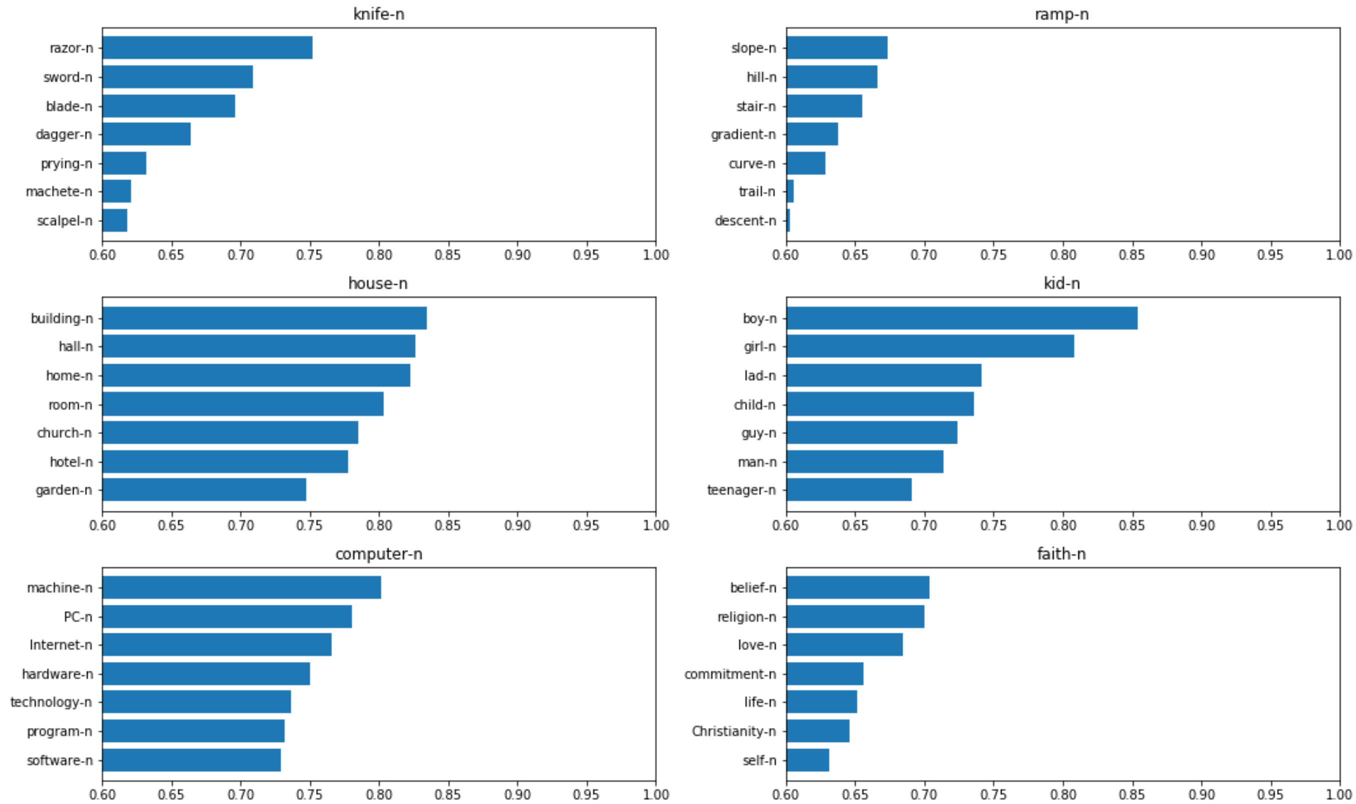

Fig. 1. Most similar nouns based on cosine similarity between embedding vectors with respect to a few selected nouns. The top seven results are shown.

TABLE III
PERFORMANCE ON STANDARD DATASETS FOR THE WORD SIMILARITY TASK EXPRESSED BY SPEARMAN'S AND PEARSON'S CORRELATION INDICES.

	Spearman's rho	**Pearson's r**
WS-353	0,326	0,361
RG-65	0,806	0,798
MC-28	0,805	0,788
SimLex-999	0,372	0,393

is rated 9.00 (SL-999) against 9.10 (WS-353) while the pair `clothes - closet` 1.96 (SL-999) against 8.00 (WS-353).

The results are shown in Table III where it can be seen that the results reach appreciable levels, especially considering that no optimization whatsoever has been applied in order to fine tune the performance on the specific datasets.

State of the art performance on similarity tasks in the above datasets can be found on the website of the Association for Computational Linguistics[3]. While being definitely preliminary, the obtained results exceed our expectations and motivate us to further refine and investigate the approach proposed here.

[3]https://aclweb.org/aclwiki/Similarity_(State_of_the_art)

TABLE IV
MOST SIGNIFICANT FEATURES RESULTING FROM THE ELEMENT-WISE PRODUCT OF THE EMBEDDINGS REPRESENTING KNIFE-N AND RAZOR-N.

Index	**Value**	**Adj**
4181	64930.269	sharp-j
810	2795.800	blunt-j
2105	1660.436	good-j
4618	1399.952	thin-j
2678	1143.005	keen-j
1462	987.955	disposable-j
1885	955.539	fine-j
1913	772.519	flat-j

C. Linguistic Word Embeddings Explainability

Even though similarity performance on standard datasets is clearly lacking behind specifically developed approaches that incorporate the latest developments in learned word embeddings and/or the incorporation of structured knowledge (eg. WordNet), the main characteristic of the proposed model is its complete interpretability.

For example, if we wanted to investigate why `knife-n` is considered similar to `razor-n`, we could perform the element-wise product between the two vectors to highlight common features. The result is shown in Table IV.

The table shows the values of the product vector with values in descending order along with the corresponding adjective and index in the Adjectives vocabulary. The reported features actually match properties commonly used to describe both objects.

VII. Conclusions

In this work, we propose a novel real-valued vector representation for word embeddings that enjoys the properties of learned word embeddings while being human understandable. The proposed approach is built on the Distributional Memory tensor [22] developing a vectorial representation where each component (dimension) corresponds to an adjective in the English language. The values of the components are proportional to the co-occurences of the noun with the adjectives, as detected in the corpus on which the DM tensor was trained.

The proposed explicit linguistic word embeddings showed decent performance on standard datasets for word similarity, and demonstrated to be able to perform tasks commonly done by learned and non interpretable word embeddings. Although preliminary, the obtained results exceed our expectations and motivate us to further refine and investigate the proposed approach, that is in line with the current research thrust aimed at developing explainable models for Artificial Intelligence.

Distributed semantic models have been often compared to other models built upon structured lexical resources like Lexica, Thesauri or WordNet [28], [29] leading to contrasting conclusions. As future work, we plan to investigate the integration of such resources in our model, conveniently weighting the links between the items which can be found within the structures.

Acknowledgment

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] G. Ferrin, L. Snidaro, and G. L. Foresti, "Describing capability through lexical semantics exploitation: Foundational arguments," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 1912–1916.

[2] M. Uemura, N. Orita, N. Okazaki, and K. Inui, "Toward the automatic extraction of knowledge of usable goods," in *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Oral Papers*, 2016, pp. 277–285.

[3] J. R. Hobbs and M. del Rey, "Word meaning and world knowledge," *Semantics: An international handbook of natural language meaning*, vol. 1, pp. 740–761, 2011.

[4] L. Wittgenstein, *Philosophical Investigations*. Oxford: Blackwell Publishers, 2001.

[5] J. R. Firth, "A synopsis of linguistic theory, 1930-1955," *Studies in linguistic analysis*, pp. 1–32, 1957.

[6] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[7] P. Aquaviva, A. Lenci, C. Paradis, and I. Raffaelli, *Models of lexical meaning*. De Gruyter Mouton, 2017.

[8] R. Richie, W. Zou, and S. Bhatia, "Semantic representations extracted from large language corpora predict high-level human judgment in seven diverse behavioral domains," 2019.

[9] A. Lenci, "Distributional models of word meaning," *Annual Review of Linguistics*, vol. 4, no. 1, pp. 151–171, 2018.

[10] G. K. Zipf, *Selected studies of the principle of relative frequency in language*. Harvard university press, 1932.

[11] 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.

[12] R. R. Hoffman and G. Klein, "Explaining explanation, part 1: theoretical foundations," *IEEE Intelligent Systems*, vol. 32, no. 3, pp. 68–73, 2017.

[13] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L119, pp. 1–88, May 2016. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC

[14] A. Chander and R. Srinivasan, "Evaluating explanations by cognitive value," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2018, pp. 314–328.

[15] L. K. Şenel, I. Utlu, V. Yücesoy, A. Koc, and T. Cukur, "Semantic structure and interpretability of word embeddings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1769–1779, 2018.

[16] M. Faruqui and C. Dyer, "Non-distributional word vector representations," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, vol. 2, 2015, pp. 464–469.

[17] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," in *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, 2016, pp. 30–35.

[18] M. Sahlgren and A. Lenci, "The effects of data size and frequency range on distributional semantic models," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 975–980.

[19] G. Lapesa and S. Evert, "Large-scale evaluation of dependency-based dsms: Are they worth the effort?" in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, vol. 2, 2017, pp. 394–400.

[20] P. Mandera, E. Keuleers, and M. Brysbaert, "Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: A review and empirical validation," *Journal of Memory and Language*, vol. 92, pp. 57–78, 2017.

[21] M. Baroni and A. Lenci, "One distributional memory, many semantic spaces," in *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*. Association for Computational Linguistics, 31 March 2009, pp. 1–8.

[22] ——, "Distributional memory: A general framework for corpus-based semantics," *Computational Linguistics*, vol. 36, no. 4, pp. 673–721, 2010.

[23] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[24] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Communications of the ACM*, vol. 8, no. 10, pp. 627–633, 1965.

[25] G. A. Miller and W. G. Charles, "Contextual correlates of semantic similarity," *Language and cognitive processes*, vol. 6, no. 1, pp. 1–28, 1991.

[26] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," *ACM Transactions on information systems*, vol. 20, no. 1, pp. 116–131, 2002.

[27] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015.

[28] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[29] C. Fellbaum, *WordNet: An Electronic Lexical Database*, ser. Language, speech, and communication. MIT Press, 1998.