Encoding syntactic dependencies by vector permutation

Pierpaolo Basile
Dept. of Computer Science
University of Bari
Via Orabona, 4
I-70125, Bari (ITALY)
basilepp@di.uniba.it

Annalina Caputo
Dept. of Computer Science
University of Bari
Via Orabona, 4
I-70125, Bari (ITALY)
acaputo@di.uniba.it

Giovanni Semeraro
Dept. of Computer Science
University of Bari
Via Orabona, 4
I-70125, Bari (ITALY)
semeraro@di.uniba.it

Abstract

Distributional approaches are based on a simple hypothesis: the meaning of a word can be inferred from its usage. The application of that idea to the vector space model makes possible the construction of a WordSpace in which words are represented by mathematical points in a geometric space. Similar words are represented close in this space and the definition of "word usage" depends on the definition of the context used to build the space, which can be the whole document, the sentence in which the word occurs, a fixed window of words, or a specific syntactic context. However, in its original formulation WordSpace can take into account only one definition of context at a time. We propose an approach based on vector permutation and Random Indexing to encode several syntactic contexts in a single WordSpace. Moreover, we propose some operations in this space and report the results of an evaluation performed using the GEMS 2011 Shared Evaluation data.

1 Background and motivation

Distributional approaches usually rely on the WordSpace model (Schütze, 1993). An overview can be found in (Sahlgren, 2006). This model is based on a vector space in which points are used to represent semantic concepts, such as words.

The core idea behind WordSpace is that words and concepts are represented by points in a mathematical space, and this representation is learned from text in such a way that concepts with similar or related meanings are near to one another in that space (geometric metaphor of meaning). The semantic similarity between concepts can be represented as proximity in an *n*-dimensional space. Therefore, the main feature of the geometric metaphor of meaning is not that meanings can be represented as locations in a semantic space, but rather that similarity between word meanings can be expressed in spatial terms, as proximity in a high-dimensional space.

One of the great virtues of WordSpaces is that they make very few language-specific assumptions, since just tokenized text is needed to build semantic spaces. Even more important is their independency of the quality (and the quantity) of available training material, since they can be built by exploiting an entirely unsupervised distributional analysis of free text. Indeed, the basis of the WordSpace model is the *distributional hypothesis* (Harris, 1968), according to which the meaning of a word is determined by the set of textual *contexts* in which it appears. As a consequence, in distributional models words can be represented as vectors built over the observable *contexts*. This means that words are semantically related as much as they are represented by similar vectors. For example, if "basketball" and "tennis" occur frequently in the same context, say after "play", they are semantically related or similar according to the distributional hypothesis.

Since co-occurrence is defined with respect to a context, co-occurring words can be stored into matrices whose rows represent the terms and columns represent contexts. More specifically, each row corresponds to a vector representation of a word. The strength of the semantic association between words can be computed by using cosine similarity.

A weak point of distributional approaches is that they are able to encode only one definition of context at a time. The type of semantics represented in WordSpace depends on the context. If we choose documents as context we obtain a semantics different from the one we would obtain by selecting sentences as context. Several approaches have investigated the above mentioned problem: (Baroni and Lenci, 2010) use a representation based on third-order tensors and provide a general framework for distributional semantics in which it is possible to represent several aspects of meaning using a single data structure. (Sahlgren et al., 2008) adopt vector permutations as a means to encode order in WordSpace, as described in Section 2. BEAGLE (Jones and Mewhort, 2007) is a very well-known method to encode word order and context information in WordSpace. The drawback of the BEAGLE model is that it relies on a complex model to build vectors which is computational expensive. This problem is solved by (De Vine and Bruza, 2010) in which the authors propose an approach similar to BEAGLE, but using a method based on Circular Holographic Reduced Representations to compute vectors.

All these methods tackle the problem of representing word order in WordSpace, but they do not take into account syntactic context. A valuable attempt in this direction is described in (Padó and Lapata, 2007). In this work, the authors propose a method to build WordSpace using information about syntactic dependencies. In particular, they consider syntactic dependencies as context and assign different weights to each kind of dependency. Moreover, they take into account the distance between two words into the graph of dependencies. The results obtained by the authors support our hypothesis that syntactic information can be useful to produce effective WordSpace. Nonetheless, their methods are not able to directly encode syntactic dependencies into the space.

This work aims to provide a simple approach to encode syntactic relations dependencies directly into the WordSpace, dealing with both the scalability problem and the possibility to encode several context information. To achieve that goal, we developed a strategy based on Random Indexing and vector permutations. Moreover, this strategy opens new possibilities in the area of semantic composition as a result of the inherent capability of encoding relations between words.

The paper is structured as follows. Section 2 describes Random Indexing, the strategy for building our WordSpace, while details about the method used to encode syntactic dependencies are reported in Section 3. Section 4 describes the formal definition of some operations over the WordSpace and shows a first attempt to define a model for semantic composition. Finally, the results of the evaluation performed using the GEMS 2011 Shared Evaluation data[1] is presented in Section 5, while conclusions are reported in Section 6.

2 Random Indexing

We exploit Random Indexing (RI), introduced by Kanerva (Kanerva, 1988), for creating a WordSpace. This technique allows us to build a WordSpace with no need for (either term-document or term-term) matrix factorization, because vectors are inferred by using an incremental strategy. Moreover, it allows to solve efficiently the problem of reducing dimensions, which is one of the key features used to uncover the "latent semantic dimensions" of a word distribution.

RI is based on the concept of Random Projection according to which high dimensional vectors chosen randomly are "nearly orthogonal".

Formally, given an $n \times m$ matrix A and an $m \times k$ matrix R made up of k m-dimensional random vectors, we define a new $n \times k$ matrix B as follows:

$$B^{n,k} = A^{n,m} \cdot R^{m,k} \quad k << m \tag{1}$$

The new matrix B has the property to preserve the distance between points. This property is known as Johnson-Lindenstrauss lemma: if the distance between two any points of A is d, then the distance d_r between the corresponding points in B will satisfy the property that $d_r = c \cdot d$. A proof of that property is reported in (Dasgupta and Gupta, 1999).

Specifically, RI creates a WordSpace in two steps (in this case we consider the document as context):

[1] Available on line: http://sites.google.com/site/geometricalmodels/shared-evaluation

1. a context vector is assigned to each document. This vector is sparse, high-dimensional and ternary, which means that its elements can take values in {-1, 0, 1}. A context vector contains a small number of randomly distributed non-zero elements, and the structure of this vector follows the hypothesis behind the concept of Random Projection;

2. context vectors are accumulated by analyzing terms and documents in which terms occur. In particular, the semantic vector for a term is computed as the sum of the context vectors for the documents which contain that term. Context vectors are multiplied by term occurrences.

Formally, given a collection of documents D whose vocabulary of terms is V (we denote with $dim(D)$ and $dim(V)$ the dimension of D and V, respectively) the above steps can be formalized as follows:

1. $\forall d_i \in D$, $i = 0,..,dim(D)$ we built the correspondent randomly generated context vector as:
$$\overrightarrow{r_j} = (r_{i1}, ..., r_{in}) \tag{2}$$
where $n \ll \dim(D)$, $r_{i*} \in \{-1, 0, 1\}$ and $\overrightarrow{r_j}$ contains only a small number of elements different from zero;

2. the WordSpace is made up of all term vectors $\overrightarrow{t_j}$ where:
$$\overrightarrow{t_j} = tf_j \sum_{\substack{d_i \in D \\ t_j \in d_i}} \overrightarrow{r_i} \tag{3}$$
and tf_j is the number of occurrences of t_j in d_i;

By considering a fixed window W of terms as context, the WordSpace is built as follows:

1. a context vector is assigned to each term;

2. context vectors are accumulated by analyzing terms in which terms co-occur in a window W. In particular, the semantic vector for each term is computed as the sum of the context vectors for terms which co-occur in W.

It is important to point out that the classical RI approach can handle only one context at a time, such as the whole document or the window W.

A method to add information about context in RI is proposed in (Sahlgren et al., 2008). The authors describe a strategy to encode word order in RI by the permutation of coordinates in random vector. When the coordinates are shuffled using a random permutation, the resulting vector is nearly orthogonal to the original one. That operation corresponds to the generation of a new random vector. Moreover, by applying a predetermined mechanism to obtain random permutations, such as elements rotation, it is always possible to reconstruct the original vector using the reverse permutations. By exploiting this strategy it is possible to obtain different random vectors for each context[2] in which the term occurs. Let us consider the following example “The cat eats the mouse”. To encode the word order for the word “cat” using a context window $W = 3$, we obtain:

$$\begin{aligned} < cat >= (\Pi^{-1} the) + (\Pi^{+1} eat) + \\ + (\Pi^{+2} the) + (\Pi^{+3} mouse) \end{aligned} \tag{4}$$

where $\Pi^n x$ indicates a rotation by n places of the elements in the vector x. Indeed, the rotation is performed by n right-shifting steps.

3 Encoding syntactic dependencies

Our idea is to encode syntactic dependencies, instead of words order, in the WordSpace using vector permutations.

A syntactic dependency between two words is defined as:
$$dep(head, dependent) \tag{5}$$
where dep is the syntactic link which connects the $dependent$ word to the $head$ word. Generally speaking, $dependent$ is the modifier, object or complement, while $head$ plays a key role in determining the behavior of the link. For example, $subj(eat, cat)$ means that “cat” is the subject of “eat”. In that case the $head$ word is “eat”, which plays the role of verb.

The key idea is to assign a permutation function to each kind of syntactic dependencies. Formally,

[2]In the case in point the context corresponds to the word order

let D be the set of all dependencies that we take into account. The function $f : D \rightarrow \Pi$ returns a schema of vector permutation for each $dep \in D$. Then, the method adopted to construct a semantic space that takes into account both syntactic dependencies and Random Indexing can be defined as follows:

1. a context vector is assigned to each term, as described in Section 2 (Random Indexing);

2. context vectors are accumulated by analyzing terms which are linked by a dependency. In particular the semantic vector for each term t_i is computed as the sum of the permuted context vectors for the terms t_j which are dependents of t_i and the inverse-permuted vectors for the terms t_j which are heads of t_i. The permutation is computed according to f. If $f(d) = \Pi^n$ the inverse-permutation is defined as $f^{-1}(d) = \Pi^{-n}$: the elements rotation is performed by n left-shifting steps.

Adding permuted vectors to the head word and inverse-permuted vectors to the corresponding dependent word allows to encode the information about both heads and dependents into the space. This approach is similar to the one investigated by (Cohen et al., 2010) to encode relations between medical terms.

To clarify, we provide an example. Given the following definition of f:

$$f(subj) = \Pi^{+3} \qquad f(obj) = \Pi^{+7} \tag{6}$$

and the sentence "The cat eats the mouse", we obtain the following dependencies:

$$\begin{array}{cc} det(the, cat) & subj(eat, cat) \\ obj(eat, mouse) & det(the, mouse) \end{array} \tag{7}$$

The semantic vector for each word is computed as:

- *eat*:
$$< eat >= (\Pi^{+3} cat) + (\Pi^{+7} mouse) \tag{8}$$
- *cat*:
$$< cat >= (\Pi^{-3} eat) \tag{9}$$
- *mouse*:
$$< mouse >= (\Pi^{-7} eat) \tag{10}$$

In the above examples, the function f does not consider the dependency det.

4 Query and vector operations

In this section, we propose two types of queries that allow us to compute semantic similarity between two words exploiting syntactic dependencies encoded in our space. Before defining query and vector operations, we introduce a small set of notations:

- R denotes the original space of random vectors generated during the WordSpace construction;
- S is the space of terms built using our strategy;
- $r_{t_i} \in R$ denotes the random vector of the term t_i;
- $s_{t_i} \in S$ denotes the semantic vector of the term t_i;
- $sim(v_1, v_2)$ denotes the similarity between two vectors; in our approach we adopt cosine similarity;
- Π^{dep} is the permutation returned from $f(dep)$. Π^{-dep} is the inverse-permutation.

The first family of queries is $dep(t_i, ?)$. The idea is to find all the dependents which are in relation with the $head$ t_i, given the dependency dep. The query can be computed as follows:

1. retrieve the vector s_{t_i} from S;
2. for each $r_{t_j} \in R$ compute the similarity between s_{t_i} and $< \Pi^{dep} r_{t_j} >$:
$$sim(s_{t_i}, < \Pi^{dep} r_{t_j} >);$$
3. rank in descending order all t_j according to the similarity computed in step 2.

The idea behind this operation is to compute how each possible dependent t_j contributes to the vector t_i, which is the sum of all the dependents related to t_i. It is important to note that we must first apply the permutation to each r_{t_j} in order to take into account the dependency relation (context). This operation has a semantics different from performing the query by applying first the inverse permutation to t_i in R and then computing the similarity with respect to all the vectors t_j in S. Indeed, the last approach would compute how the head t_i contributes to the vector t_j, which differs from the goal of our query.

Using the same approach it is possible to compute the query $dep(?, t_j)$, in which we want to search all the *head*s related to the *dependent* t_j fixed the dependency *dep*. In detail:

1. retrieve the vector s_{t_j} from S;
2. for each $r_{t_i} \in R$ compute the similarity between s_{t_j} and the inverse-permutation of r_{t_i}, $< \Pi^{-dep} r_{t_i} >$: $sim(s_{t_j}, < \Pi^{-dep} r_{t_i} >)$;
3. rank in descending order all t_i according to the similarity computed in step 2.

In this second query, we compute how the inverse-permutation of each t_i (head) affects the vector $s_{t_j} \in S$. In the following sub-section we provide some initial idea about semantic composition.

4.1 Compositional semantics

Distributional approaches represent words in isolation and they are typically used to compute similarities between words. They are not able to represent complex structures such as phrases or sentences. In some applications, such as Question Answering and Text Entailment, representing text by single words is not enough. These applications would benefit from the composition of words in more complex structures. The strength of our approach lies on the capability of codify syntactic relations between words overcoming the "word isolation" issue.

A lot of recent work argue that tensor product ($\otimes$) could be useful to combine word vectors. In (Widdows, 2008) some preliminary investigations about product and tensor product are provided, while an interesting work by Clark and Pulman (Clark and Pulman, 2007) proposes an approach to combine symbolic and distributional models. The main idea is to use tensor product to combine these two aspects, but the authors do not describe a method to represent symbolic features, such as syntactic dependencies. Conversely, our approach is able to encode syntactic information directly into the distributional model. The authors in (Clark and Pulman, 2007) propose a strategy to represent a sentence like "man reads magazine" by tensor product:

$$man \otimes subj \otimes read \otimes obj \otimes magazine \quad (11)$$

They also propose a solid model for compositionality, but they do not provide a strategy to represent symbolic relations, such as $subj$ and obj. They wrote: "How to obtain vectors for the dependency relations - subj, obj, etc. - is an open question". We believe that our approach can tackle this problem by encoding the dependency directly in the space, because each semantic vector in our space contains information about syntactic roles.

The representation based on tensor product is useful to compute sentence similarity. Given the previous sentence and the following one "woman browses newspaper", we want to compute the similarity between the two sentences. The sentence "woman browses newspaper", using the compositional model, is represented by:

$$woman \otimes subj \otimes browse \otimes obj \otimes newspaper \quad (12)$$

Computing the similarity of two representations by inner product is a complex task, but exploiting the following property of the tensor product:

$$(w_1 \otimes w_2) \cdot (w_3 \otimes w_4) = (w_1 \cdot w_3) \times (w_2 \cdot w_4) \quad (13)$$

the similarity between two sentences can be computed by taking into account the pairs in each dependency and multiplying the inner products as follows:

$$\begin{aligned} man \cdot woman \times read \cdot browse \times \\ \times magazine \cdot newspaper \end{aligned} \quad (14)$$

According to the property above mentioned, we can compute the similarity between sentences without using the tensor product. However, some open questions arise. This simple compositional strategy allows to compare sentences which have similar dependency trees. For example, the sentence "the dog bit the man" cannot can be compared to "the man was bitten by the dog". This problem can be easily solved by identifying active and passive forms of a verb. When two sentences have different trees, Clark and Pulman (Clark and Pulman, 2007) propose to adopt the *convolution kernel* (Haussler, 1999). This strategy identifies all the possible ways of decomposing the two trees, and sums up the similarities between all the pairwise decompositions. It is important to point out that, in a more recent work, Clark et al. (Clark et al., 2008) propose a model based on (Clark and Pulman, 2007) combined with a compositional theory for grammatical types, known as Lambek's pregroup semantics, which is able to take into account grammar structures. It is important to note that this strategy is not able to encode grammatical roles into the WordSpace. This peculiarity makes our approach completely different. In the following section we provide some examples of compositionality.

5 Evaluation

The goal of the evaluation is twofold: proving the capability of our approach by means of some examples and providing results of the evaluation exploiting the "GEMS 2011 Shared Evaluation", in particular the compositional semantics dataset. We propose two semantic spaces built from two separate corpora using our strategy. To achieve the first goal we provide several examples for each family of queries described in Section 4. Concerning the second goal, we evaluate our approach to compositional semantics using the dataset proposed by Mitchell and Lapata (Mitchell and Lapata, 2010), which is part of the "GEMS 2011 Shared Evaluation". The dataset is a list of two pairs of adjective-noun combinations or verb-object combinations or compound nouns. Humans rated pairs of combinations according to similarity. The dataset contains 5,833 rates which range from 1 to 7. Examples of pairs follow:

support offer help provide 7

old person right hand 1

where the similarity between offer-support and provide-help (verb-object) is higher than the one between old-person and right-hand (adjective-noun). As suggested by the authors, the goal of the evaluation is to compare the system performace against humans scores by means of Spearman correlation.

5.1 System setup

The system is implemented in Java and relies on some portions of code publicly available in the Semantic Vectors package (Widdows and Ferraro, 2008). For the evaluation of the system, we build two separate WordSpaces using the following corpora: ukWaC (Baroni et al., 2009) and TASA. ukWaC contains 2 billion words and it is constructed from the Web by limiting the crawling to the .uk domain and using medium-frequency words from the BNC corpus as seeds. We use only a portion of ukWaC corpus consisting of 7,025,587 sentences (about 220,000 documents). The TASA corpus (compiled by Touchstone Applied Science Associates) was kindly made available to us by Prof. Thomas Landauer from the University of Colorado. The TASA corpus contains a collection of English texts that is approximately equivalent to what the average college-level student has read in his/her lifetime. The TASA corpus consists of about 800,000 sentences.

To extract syntactic dependencies, we adopt MINIPAR[3] (Lin, 2003). MINIPAR is an efficient English parser, which is suitable for parsing a large amount of data. The total amount of extracted dependencies is about 112,500,000 for ukWaC and 8,850,000 for TASA.

Our approach involves some parameters. We set the random vector dimension to 4,000 and the number of non-zero elements in the random vector equal to 10. We restrict the WordSpace to the 40,000 most frequent words[4]. Another parameter is the set of dependencies that we take into account. In this preliminary investigation we consider the four dependencies described in Table 1, that reports also the kind of permutation[5] applied to vectors.

5.2 Results

In this section we report some results of queries performed in ukWaC and TASA corpus.

Table 2 and Table 3 report the results respectively for the queries $dep(t_i, ?)$ and $dep(?, t_j)$. The effects of encoding syntactic information is clearly visible, as can be inferred by results in the tables. Moreover, the results with the two corpora are different, as expected, but in many cases the first result of the query is the same.

Our space can be also exploited to perform classical queries in which we want to find "similar" words. Tables 4 and 5 report results for TASA and ukWaC

[3]MINIPAR is available at http://webdocs.cs.ualberta.ca/~lindek/minipar.htm

[4]Word frequency is computed taking into account the selected dependencies.

[5]The number of rotations is randomly chosen.

Dependency	**Description**	**Permutation**
obj	object of verbs	Π^{+7}
subj	subject of verbs	Π^{+3}
mod	the relationship between a word and its adjunct modifier	Π^{+11}
comp	complement	Π^{+23}

Table 1: The set of dependencies used in the evaluation.

corpus, respectively. The results obtained by similar test are not the typical results expected by classical WordSpace. In fact, in Table 5 the word most similar to "good" is "bad", because they are used in the same syntactic context, but have opposite meaning. The similarity between words in our space strongly depends on their syntactic role. For example, the words similar to "food" are all the nouns which are object/subject of the same verbs in syntactic relation with "food".

Finally, we provide the results of semantic composition. Table 6 reports the Spearman correlation between the output of our system and the mean similarity scores given by the humans. The table shows results for each types of combination: verb-object, adjective-noun and compound nouns. To perform the experiment on compound nouns, we rebuild the spaces encoding the "nn" relation provided by MINIPAR which refers to compound nouns dependency. Table 6 shows the best result obtained by Mitchell and Lapata (Mitchell and Lapata, 2008) using the same dataset. Our method is able to outperform ML_{best} and obtains very high results when adjective-noun combination is involved.

Corpus	Combination	ρ
TASA	verb-object	0.260
	adjective-noun	0.637
	compound nouns	0.341
	overall	**0.275**
ukWaC	verb-object	0.292
	adjective-noun	0.445
	compound nouns	0.227
	overall	**0.261**
-	ML_{best}	*0.190*

Table 6: GEMS 2011 Shared Evaluation results.

The experiments reported in this preliminary evaluation are only a small fraction of the experiments that are required to make a proper evaluation of the effectiveness of our semantic space and to compare it with other approaches. This will be the main focus of our future research. The obtained results seem to be encouraging and the strength of our approach, capturing syntactic relations, allows to implement several kind of queries using only one WordSpace. We believe that the real advantage of our approach, that is the possibility to represent several syntactic relations, has much room for exploration.

6 Conclusions

In this work, we propose an approach to encode syntactic dependencies in WordSpace using vector permutations and Random Indexing. In that space, a set of operations is defined, which relies on the possibility of exploiting syntactic dependencies to perform some particular queries, such as the one for retrieving all similar objects of a verb. We propose an early attempt to use that space for semantic composition of short sentences. The evaluation using the GEMS 2011 shared dataset provides encouraging results, but we believe that there are open points which deserve more investigation. We planned a deeper evaluation of our WordSpace and a more formal study about semantic composition.

Acknowledgements

This research was partially funded by MIUR (Ministero dell'Università e della Ricerca) under the contract Fondo per le Agevolazioni alla Ricerca, DM19410 "Laboratorio" di Bioinformatica per la Biodiversità Molecolare (2007-2011).

$obj(provide, ?)$				$mod(people, ?)$			
TASA		ukWaC		TASA		ukWaC	
information	0.344	information	0.351	young	0.288	young	0.736
food	0.208	service	0.260	black	0.118	with	0.360
support	0.143	you	0.176	old	0.089	other	0.223
energy	0.143	opportunity	0.141	conquered	0.086	handling	0.164
job	0.142	support	0.127	deaf	0.086	impressive	0.162

Table 2: Examples of query $dep(t_i, ?)$.

$obj(?, food)$				$mod(?, good)$			
TASA		ukWaC		TASA		ukWaC	
eat	0.604	eat	0.429	idea	0.350	practice	0.510
make	0.389	serve	0.256	place	0.320	idea	0.363
grow	0.311	provide	0.230	way	0.269	news	0.274
need	0.272	have	0.177	friend	0.246	for	0.269
store	0.161	buy	0.169	time	0.234	very	0.228

Table 3: Examples of query $dep(?, t_j)$.

food		provide		good	
food	1.000	provide	1.000	good	0.999
foods	0.698	make	0.702	best	0.498
meat	0.654	restructure	0.693	excellent	0.471
meal	0.651	ready	0.680	wrong	0.453
bread	0.606	leave	0.673	main	0.430
wheato	0.604	mean	0.672	nice	0.428
thirty_percent	0.604	work	0.672	safe	0.428
mezas	0.604	offer	0.671	new	0.428
orgy	0.604	relate	0.667	proper	0.400
chocolatebar	0.604	gather	0.667	surrounded	0.400

Table 4: Find similar words, TASA corpus.

food		provide		good	
food	1.000	provide	0.999	good	1.000
meal	0.724	offer	0.855	bad	0.603
meat	0.656	supply	0.819	best	0.545
pie	0.578	deliver	0.801	anti-discriminatory	0.507
tea	0.576	give	0.787	nice	0.478
fresh_food	0.576	contain	0.786	reflective	0.470
supper	0.556	require	0.784	brilliant	0.464
porridge	0.553	present	0.782	great	0.462
entertainment	0.533	gather	0.778	evidence-based	0.453
soup	0.532	work	0.777	unsafe	0.444

Table 5: Find similar words, ukWaC corpus.

References

M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2009. The WaCky Wide Web: A collection of very large linguistically processed Web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

S. Clark and S. Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, pages 52–55.

S. Clark, B. Coecke, and M. Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, pages 133–140.

T. Cohen, D. Widdows, R.W. Schvaneveldt, and T.C. Rindflesch. 2010. Logical leaps and quantum connectives: Forging paths through predication space. In *AAAI-Fall 2010 Symposium on Quantum Informatics for Cognitive, Social, and Semantic Processes*, pages 11–13.

S. Dasgupta and A. Gupta. 1999. An elementary proof of the Johnson-Lindenstrauss lemma. Technical report, Technical Report TR-99-006, International Computer Science Institute, Berkeley, California, USA.

L. De Vine and P. Bruza. 2010. Semantic Oscillations: Encoding Context and Structure in Complex Valued Holographic Vectors. *Quantum Informatics for Cognitive, Social, and Semantic Processes (QI 2010)*.

Z. Harris. 1968. *Mathematical Structures of Language*. New York: Interscience.

D. Haussler. 1999. Convolution kernels on discrete structures. *Technical Report UCSC-CRL-99-10*.

M.N. Jones and D.J.K. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, 114(1):1–37.

P. Kanerva. 1988. *Sparse Distributed Memory*. MIT Press.

D. Lin. 2003. Dependency-based evaluation of MINIPAR. *Treebanks: building and using parsed corpora*.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June. Association for Computational Linguistics.

J. Mitchell and M. Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*. To appear.

S. Padó and M. Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

M. Sahlgren, A. Holst, and P. Kanerva. 2008. Permutations as a means to encode order in word space. In *Proceedings of the 30th Annual Meeting of the Cognitive Science Society (CogSci'08)*.

M. Sahlgren. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Stockholm: Stockholm University, Faculty of Humanities, Department of Linguistics.

H. Schütze. 1993. Word space. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, pages 895–902. Morgan Kaufmann Publishers.

D. Widdows and K. Ferraro. 2008. Semantic Vectors: A Scalable Open Source Package and Online Technology Management Application. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

D. Widdows. 2008. Semantic vector products: Some initial investigations. In *The Second AAAI Symposium on Quantum Interaction*.