How to make words with vectors: Phrase generation in distributional semantics

Georgiana Dinu and **Marco Baroni**
Center for Mind/Brain Sciences
University of Trento, Italy
(georgiana.dinu|marco.baroni)@unitn.it

Abstract

We introduce the problem of *generation* in distributional semantics: Given a distributional vector representing some meaning, how can we generate the phrase that best expresses that meaning? We motivate this novel challenge on theoretical and practical grounds and propose a simple data-driven approach to the estimation of generation functions. We test this in a monolingual scenario (paraphrase generation) as well as in a cross-lingual setting (translation by synthesizing adjective-noun phrase vectors in English and generating the equivalent expressions in Italian).

1 Introduction

Distributional methods for semantics approximate the meaning of linguistic expressions with vectors that summarize the contexts in which they occur in large samples of text. This has been a very successful approach to lexical semantics (Erk, 2012), where semantic relatedness is assessed by comparing vectors. Recently these methods have been extended to phrases and sentences by means of composition operations (see Baroni (2013) for an overview). For example, given the vectors representing *red* and *car*, composition derives a vector that approximates the meaning of *red car*.

However, the link between language and meaning is, obviously, bidirectional: As message recipients we are exposed to a linguistic expression and we must compute its meaning (the *synthesis* problem). As message producers we start from the meaning we want to communicate (a "thought") and we must encode it into a word sequence (the *generation* problem). If distributional semantics is to be considered a proper semantic theory, then it must deal not only with synthesis (going from words to vectors), but also with generation (from vectors to words).

Besides these theoretical considerations, phrase generation from vectors has many useful applications. We can, for example, synthesize the vector representing the meaning of a phrase or sentence, and then generate alternative phrases or sentences from this vector to accomplish true *paraphrase generation* (as opposed to paraphrase detection or ranking of candidate paraphrases).

Generation can be even more useful when the source vector comes from another modality or language. Recent work on grounding language in vision shows that it is possible to represent images and linguistic expressions in a common vector-based semantic space (Frome et al., 2013; Socher et al., 2013). Given a vector representing an image, generation can be used to productively construct phrases or sentences that describe the image (as opposed to simply retrieving an existing description from a set of candidates). Translation is another potential application of the generation framework: Given a semantic space shared between two or more languages, one can compose a word sequence in one language and generate translations in another, with the shared semantic vector space functioning as interlingua.

Distributional semantics assumes a lexicon of atomic expressions (that, for simplicity, we take to be *words*), each associated to a vector. Thus, at the single-word level, the problem of generation is solved by a trivial *generation-by-synthesis* approach: Given an arbitrary target vector, "generate" the corresponding word by searching through the lexicon for the word with the closest vector to the target. This is however unfeasible for larger expressions: Given n vocabulary elements, this approach requires checking n^k phrases of length k. This becomes prohibitive already for relatively short phrases, as reasonably-sized vocabularies do not go below tens of thousands of words. The search space for 3-word phrases in a 10K-word vocabulary is already in the order of trillions. In

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 624–633,
Baltimore, Maryland, USA, June 23-25 2014.

this paper, we introduce a more direct approach to phrase generation, inspired by the work in compositional distributional semantics. In short, we revert the composition process and we propose a framework of data-induced, syntax-dependent functions that *decompose* a single vector into a vector sequence. The generated vectors can then be efficiently matched against those in the lexicon or fed to the decomposition system again to produce longer phrases recursively.

2 Related work

To the best of our knowledge, we are the first to explicitly and systematically pursue the generation problem in distributional semantics. Kalchbrenner and Blunsom (2013) use top-level, composed distributed representations of sentences to guide generation in a machine translation setting. More precisely, they condition the target language model on the composed representation (addition of word vectors) of the source language sentence.

Andreas and Ghahramani (2013) discuss the the issue of generating language from vectors and present a probabilistic generative model for distributional vectors. However, their emphasis is on *reversing* the generative story in order to derive composed meaning representations from word sequences. The theoretical generating capabilities of the methods they propose are briefly exemplified, but not fully explored or tested.

Socher et al. (2011) come closest to our target problem. They introduce a bidirectional language-to-meaning model for compositional distributional semantics that is similar in spirit to ours. However, we present a clearer decoupling of synthesis and generation and we use different (and simpler) training methods and objective functions. Moreover, Socher and colleagues do not train separate decomposition rules for different syntactic configurations, so it is not clear how they would be able to control the generation of different output structures. Finally, the potential for generation is only addressed in passing, by presenting a few cases where the generated sequence has the same syntactic structure of the input sequence.

3 General framework

We start by presenting the familiar synthesis setting, focusing on two-word phrases. We then introduce generation for the same structures. Finally, we show how synthesis and generation of longer phrases is handled by recursive extension of the two-word case. We assume a lexicon L, that is, a bi-directional look-up table containing a list of words L_w linked to a matrix L_v of vectors. Both synthesis and generation involve a trivial lexicon look-up step to retrieve vectors associated to words and *vice versa*: We ignore it in the exposition below.

3.1 Synthesis

To construct the vector representing a two-word phrase, we must compose the vectors associated to the input words. More formally, similarly to Mitchell and Lapata (2008), we define a syntax-dependent *composition* function yielding a phrase vector $\vec{p}$:

$$\vec{p} = \mathrm{f}_{\mathrm{comp}_\mathrm{R}}(\vec{u}, \vec{v})$$

where $\vec{u}$ and $\vec{v}$ are the vector representations associated to words u and v. $\mathrm{f}_{\mathrm{comp}_\mathrm{R}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ (for d the dimensionality of vectors) is a composition function specific to the syntactic relation R holding between the two words.[1]

Although we are not bound to a specific composition model, throughout this paper we use the method proposed by Guevara (2010) and Zanzotto et al. (2010) which defines composition as application of linear transformations to the two constituents followed by summing the resulting vectors: $\mathrm{f}_{\mathrm{comp}_\mathrm{R}}(\vec{u}, \vec{v}) = W_1\vec{u} + W_2\vec{v}$. We will further use the following equivalent formulation:

$$\mathrm{f}_{\mathrm{comp}_\mathrm{R}}(\vec{u}, \vec{v}) = W_R[\vec{u}; \vec{v}]$$

where $W_R \in R^{d \times 2d}$ and $[\vec{u}; \vec{v}]$ is the vertical concatenation of the two vectors (using Matlab notation). Following Guevara, we learn W_R using examples of word and phrase vectors directly extracted from the corpus (for the rest of the paper, we refer to these phrase vectors extracted non-compositionally from the corpus as *observed* vectors). To estimate, for example, the weights in the W_{AN} (adjective-noun) matrix, we use the corpus-extracted vectors of the words in tuples such as $\langle red, car, red.car\rangle$, $\langle evil, cat, evil.cat\rangle$, etc. Given a set of training examples stacked into matrices U, V (the constituent vectors) and P (the corresponding observed vectors), we estimate W_R by solving the least-squares regression problem:

[1]Here we make the simplifying assumption that all vectors have the same dimensionality, however this need not necessarily be the case.

$$\min_{W_R \in \mathbb{R}^{d \times 2d}} \|P - W_R[U;V]\| \quad (1)$$

We use the approximation of observed phrase vectors as objective because these vectors can provide direct evidence of the polysemous behaviour of words: For example, the corpus-observed vectors of *green jacket* and *green politician* reflect how the meaning of *green* is affected by its occurrence with different nouns. Moreover, it has been shown that for two-word phrases, despite their relatively low frequency, such corpus-observed representations are still difficult to outperform in phrase similarity tasks (Dinu et al., 2013; Turney, 2012).

3.2 Generation

Generation of a two-word sequence from a vector proceeds in two steps: decomposition of the phrase vectors into two constituent vectors, and search for the nearest neighbours of each constituent vector in L_v (the lexical matrix) in order to retrieve the corresponding words from L_w.

Decomposition We define a syntax-dependent *decomposition* function:

$$[\vec{u};\vec{v}] = \mathrm{f}_{\mathrm{decomp_R}}(\vec{p})$$

where $\vec{p}$ is a phrase vector, $\vec{u}$ and $\vec{v}$ are vectors associated to words standing in the syntactic relation R and $\mathrm{f}_{\mathrm{decomp_R}} : \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$.

We assume that decomposition is also a linear transformation, $W'_R \in \mathbb{R}^{2d \times d}$, which, given an input phrase vector, returns two constituent vectors:

$$\mathrm{f}_{\mathrm{decomp_R}}(\vec{p}) = W'_R \vec{p}$$

Again, we can learn from corpus-observed vectors associated to tuples of word pairs and the corresponding phrases by solving:

$$\min_{W'_R \in \mathbb{R}^{2d \times d}} \|[U;V] - W'_R P\| \quad (2)$$

If a composition function $\mathrm{f}_{\mathrm{comp_R}}$ is available, an alternative is to learn a function that can best *revert* this composition. The decomposition function is then trained as follows:

$$\min_{W'_R \in \mathbb{R}^{2d \times d}} \|[U;V] - W'_R W_R [U;V]\| \quad (3)$$

where the matrix W_R is a given composition function for the same relation R. Training with observed phrases, as in eq. (2), should be better at capturing the idiosyncrasies of the actual distribution of phrases in the corpus and it is more robust by being independent from the availability and quality of composition functions. On the other hand, if the goal is to revert as faithfully as possible the composition process and retrieve the original constituents (e.g., in a different modality or a different language), then the objective in eq. (3) is more motivated.

Nearest neighbour search We retrieve the nearest neighbours of each constituent vector $\vec{u}$ obtained by decomposition by applying a search function s:

$$\mathrm{NN}_{\vec{u}} = s(\vec{u}, L_v, t)$$

where $\mathrm{NN}_{\vec{u}}$ is a list containing the t nearest neighours of $\vec{u}$ from L_v, the lexical vectors. Depending on the task, t might be set to 1 to retrieve just one word sequence, or to larger values to retrieve t alternatives. The similarity measure used to determine the nearest neighbours is another parameter of the search function; we omit it here as we only experiment with the standard cosine measure (Turney and Pantel, 2010).[2]

3.3 Recursive (de)composition

Extension to longer sequences is straightforward if we assume binary tree representations as syntactic structures. In synthesis, the top-level vector can be obtained by applying composition functions recursively. For example, the vector of *big red car* would be obtained as: $\mathrm{f}_{\mathrm{comp_{AN}}}(\vec{big}, \mathrm{f}_{\mathrm{comp_{AN}}}(\vec{red}, \vec{car}))$, where $\mathrm{f}_{\mathrm{comp_{AN}}}$ is the composition function for adjective-noun phrase combinations. Conversely, for generation, we decompose the phrase vector with $\mathrm{f}_{\mathrm{decomp_{AN}}}$. The first vector is used for retrieving the nearest adjective from the lexicon, while the second vector is further decomposed.

In the experiments in this paper we assume that the syntactic structure is given. In Section 7, we discuss ways to eliminate this assumption.

[2]Note that in terms of computational efficiency, cosine-based nearest neighbour searches reduce to vector-matrix multiplications, for which many efficient implementations exist. Methods such as locality sensitive hashing can be used for further speedups when working with particularly large vocabularies (Andoni and Indyk, 2008).

4 Evaluation setting

In our empirical part, we focus on noun phrase generation. A noun phrase can be a single noun or a noun with one or more modifiers, where a modifier can be an adjective or a prepositional phrase. A prepositional phrase is in turn composed of a preposition and a noun phrase. We learn two composition (and corresponding decomposition) functions: one for modifier-noun phrases, trained on adjective-noun (AN) pairs, and a second one for prepositional phrases, trained on preposition-noun (PN) combinations. For the rest of this section we describe the construction of the vector spaces and the (de)composition function learning procedure.

Construction of vector spaces We test two types of vector representations. The *cbow* model introduced in Mikolov et al. (2013a) learns vector representations using a neural network architecture by trying to predict a target word given the words surrounding it. We use the word2vec software[3] to build vectors of size 300 and using a context window of 5 words to either side of the target. We set the sub-sampling option to 1e-05 and estimate the probability of a target word with the negative sampling method, drawing 10 samples from the noise distribution (see Mikolov et al. (2013a) for details). We also implement a standard *count*-based bag-of-words distributional space (Turney and Pantel, 2010) which counts occurrences of a target word with other words within a symmetric window of size 5. We build a 300Kx300K symmetric co-occurrence matrix using the top most frequent words in our source corpus, apply positive PMI weighting and Singular Value Decomposition to reduce the space to 300 dimensions. For both spaces, the vectors are finally normalized to unit length.[4]

For both types of vectors we use 2.8 billion tokens as input (ukWaC + Wikipedia + BNC). The Italian language vectors for the cross-lingual experiments of Section 6 were trained on 1.6 billion tokens from itWaC.[5] A word token is a wordform + POS-tag string. We extract both word vectors and the observed phrase vectors which are required for the training procedures. We sanity-check the two spaces on MEN (Bruni et al., 2012), a 3,000 items word similarity data set. *cbow* significantly outperforms *count* (0.80 vs. 0.72 Spearman correlations with human judgments). *count* performance is consistent with previously reported results.[6]

(De)composition function training The training data sets consist of the 50K most frequent $\langle u, v, p\rangle$ tuples for each phrase type, for example, $\langle red, car, red.car\rangle$ or $\langle in, car, in.car\rangle$.[7] We concatenate $\vec{u}$ and $\vec{v}$ vectors to obtain the $[U; V]$ matrix and we use the observed $\vec{p}$ vectors (e.g., the corpus vector of the *red.car* bigram) to obtain the phrase matrix P. We use these data sets to solve the least squares regression problems in eqs. (1) and (2), obtaining estimates of the composition and decomposition matrices, respectively. For the decomposition function in eq. (3), we replace the observed phrase vectors with those composed with $\mathrm{f}_{\mathrm{comp}_\mathrm{R}}(\vec{u}, \vec{v})$, where $\mathrm{f}_{\mathrm{comp}_\mathrm{R}}$ is the previously estimated composition function for relation R.

Composition function performance Since the experiments below also use composed vectors as input to the generation process, it is important to provide independent evidence that the composition model is of high quality. This is indeed the case: We tested our composition approach on the task of retrieving *observed* AN and PN vectors, based on their composed vectors (similarly to Baroni and Zamparelli (2010), we want to retrieve the observed *red.car* vector using $\mathrm{f}_{\mathrm{comp}_\mathrm{AN}}(red, car)$). We obtain excellent results, with minimum accuracy of 0.23 (chance level <0.0001). We also test on the AN-N paraphrasing test set used in Dinu et al. (2013) (in turn adapting Turney (2012)). The dataset contains 620 ANs, each paired with a single-noun paraphrase (e.g., *false belief/fallacy*, *personal appeal/charisma*). The task is to rank all nouns in the lexicon by their similarity to the phrase, and return the rank of the correct paraphrase. Results are reported in the first row of Table 1. To facilitate comparison, we search, like Dinu et al., through a vocabulary containing the 20K most frequent nouns. The *count* vectors results are similar to those reported by Dinu and colleagues for the same model, and with *cbow* vec-

[3]Available at `https://code.google.com/p/word2vec/`

[4]The parameters of both models have been chosen without specific tuning, based on their observed stable performance in previous independent experiments.

[5]Corpus sources: `http://wacky.sslmit.unibo.it`, `http://www.natcorp.ox.ac.uk`

[6]See Baroni et al. (2014) for an extensive comparison of the two types of vector representations.

[7]For PNs, we ignore determiners and we collapse, for example, *in.the.car* and *in.car* occurrences.

Input	Output	cbow	count
A∘N	**N**	11	171
N	**A, N**	67,29	204,168

Table 1: Median rank on the AN-N set of Dinu et al. (2013) (e.g., *personal appeal/charisma*). First row: the A and N are composed and the closest N is returned as a paraphrase. Second row: the N vector is *decomposed* into A and N vectors and their nearest (POS-tag consistent) neighbours are returned.

Input	Output	cbow	count
A.N	**A, N**	0.36,0.61	0.20,0.41
P.N	**P, N**	0.93,0.79	0.60,0.57
A∘N	**A, N**	1.00,1.00	0.86,0.99
P∘N	**P, N**	1.00,1.00	1.00,1.00

Table 2: Accuracy of generation models at retrieving (at rank 1) the constituent words of adjective-noun (AN) and preposition-noun (PN) phrases. Observed (**A.N**) and composed representations (**A∘N**) are decomposed with observed- (eq. 2) and composed-trained (eq. 3) functions respectively.

tors we obtain a median rank that is considerably higher than that of the methods they test.

5 Noun phrase generation

5.1 One-step decomposition

We start with testing one-step decomposition by generating two-word phrases. A first straightforward evaluation consists in decomposing a phrase vector into the correct constituent words. For this purpose, we randomly select (and consequently remove) from the training sets 200 phrases of each type (AN and PN) and apply decomposition operations to 1) their corpus-observed vectors and 2) their composed representations. We generate two words by returning the nearest neighbours (with appropriate POS tags) of the two vectors produced by the decomposition functions. Table 2 reports generation accuracy, i.e., the proportion of times in which we retrieved the correct constituents. The search space consists of the top most frequent 20K nouns, 20K adjectives and 25 prepositions respectively, leading to chance accuracy $<$0.0001 for nouns and adjectives and $<$0.05 for prepositions. We obtain relatively high accuracy, with *cbow* vectors consistently outperforming *count* ones. Decomposing composed rather than observed phrase representations is easier, which is to be expected given that composed representations are obtained with a simpler, linear model. Most of the errors consist in generating synonyms (*hard case→difficult case*, *true cost → actual cost*) or related phrases (*stereo speakers→omni-directional sound*).

Next, we use the AN-N dataset of Dinu and colleagues for a more interesting evaluation of one-step decomposition. In particular, we reverse the original paraphrasing direction by attempting to generate, for example, *personal charm* from *charisma*. It is worth stressing the nature of the paraphrase-by-generation task we tackle here and in the next experiments. Compositional distributional semantic systems are often evaluated on phrase and sentence paraphrasing data sets (Blacoe and Lapata, 2012; Mitchell and Lapata, 2010; Socher et al., 2011; Turney, 2012). However, these experiments assume a pre-compiled list of candidate paraphrases, and the task is to rank correct paraphrases above foils (paraphrase ranking) or to decide, for a given pair, if the two phrases/sentences are mutual paraphrases (paraphrase detection). Here, instead, we do not assume a given set of candidates: For example, in N→AN paraphrasing, any of $20K^2$ possible combinations of adjectives and nouns from the lexicon could be generated. This is a much more challenging task and it paves the way to more realistic applications of distributional semantics in generation scenarios.

The median ranks of the gold A and N of the Dinu set are shown in the second row of Table 1. As the top-generated noun is almost always, uninterestingly, the input one, we return the next noun. Here we report results for the more motivated corpus-observed training of eq. (2) (unsurprisingly, using composed-phrase training for the task of decomposing *single* nouns leads to lower performance).

Although considerably more difficult than the previous task, the results are still very good, with median ranks under 100 for the *cbow* vectors (random median rank at 10K). Also, the dataset provides only one AN paraphrase for each noun, out of many acceptable ones. Examples of generated phrases are given in Table 3. In addition to generating topically related ANs, we also see nouns disambiguated in different ways than intended in

Input	Output	Gold
reasoning	deductive thinking	abstract thought
jurisdiction	legal authority	legal power
thunderstorm	thundery storm	electrical storm
folk	local music	common people
superstition	old-fashioned religion	superstitious notion
vitriol	political bitterness	sulfuric acid
zoom	fantastic camera	rapid growth
religion	religious religion	religious belief

Table 3: Examples of generating ANs from Ns using the data set of Dinu et al. (2013).

the gold standard (for example *vitriol* and *folk* in Table 3). Other interesting errors consist of decomposing a noun into two words which both have the same meaning as the noun, generating for example *religion → religious religions*. We observe moreover that sometimes the decomposition reflects selectional preference effects, by generating adjectives that denote typical properties of the noun to be paraphrased (e.g., *animosity* is a *(political, personal,...) hostility* or a *fridge* is a *(big, large, small,...) refrigerator*). This effect could be exploited for tasks such as property-based concept description (Kelly et al., 2012).

5.2 Recursive decomposition

We continue by testing generation through *recursive* decomposition on the task of generating noun-preposition-noun (NPN) paraphrases of adjective-nouns (AN) phrases. We introduce a dataset containing 192 AN-NPN pairs (such as *pre-election promises→ promises before election*), which was created by the second author and additionally corrected by an English native speaker. The data set was created by analyzing a list of randomly selected frequent ANs. 49 further ANs (with adjectives such as *amazing* and *great*) were judged not NPN-paraphrasable and were used for the experiment reported in Section 7. The paraphrased subset focuses on preposition diversity and on including prepositions which are rich in semantic content and relevant to paraphrasing the AN. This has led to excluding *of*, which in most cases has the purely syntactic function of connecting the two nouns. The data set contains the following 14 prepositions: *after, against, at, before, between, by, for, from, in, on, per, under, with, without*.[8]

NPN phrase generation involves the application of two decomposition functions. In the first step we decompose using the modifier-noun rule ($\mathrm{f}_{\mathrm{decomp}_{\mathrm{AN}}}$). We generate a noun from the head slot vector and the "adjective" vector is further decomposed using $\mathrm{f}_{\mathrm{decomp}_{\mathrm{PN}}}$ (returning the top noun which is not identical to the previously generated one). The results, in terms of top 1 accuracy and median rank, are shown in Table 4. Examples are given in Table 5.

For observed phrase vector training, accuracy and rank are well above chance for all constituents (random accuracy 0.00005 for nouns and 0.04 for prepositions, corresponding median ranks: 10K, 12). Preposition generation is clearly a more difficult task. This is due at least in part to their highly ambiguous and broad semantics, and the way in which they interact with the nouns. For example, *cable through ocean* in Table 5 is a reasonable paraphrase of *undersea cable* despite the gold preposition being *under*. Other than several cases which are acceptable paraphrases but not in the gold standard, phrases related in meaning but not synonymous are the most common error (*overcast skies → skies in sunshine*). We also observe that often the A and N meanings are not fully separated when decomposing and "traces" of the adjective or of the original noun meaning can be found in *both* generated nouns (for example *nearby school → schools after school*). To a lesser degree, this might be desirable as a disambiguation-in-context effect as, for example, in *underground cavern*, *in secret* would not be a context-appropriate paraphrase of *underground*.

6 Noun phrase translation

This section describes preliminary experiments performed in a cross-lingual setting on the task of composing English AN phrases and generating Italian translations.

Creation of cross-lingual vector spaces A common semantic space is required in order to map words and phrases across languages. This problem has been extensively addressed in the bilingual lexicon acquisition literature (Haghighi et al., 2008; Koehn and Knight, 2002). We opt for a very simple yet accurate method (Klementiev et al., 2012; Rapp, 1999) in which a bilingual dictionary is used to identify a set of shared dimensions across spaces and the vectors of both languages are projected into the subspace defined by these (Subspace Projection - SP). This method is applicable to *count*-type vector spaces, for which the dimen-

[8]This dataset is available at `http://clic.cimec.unitn.it/composes`

Input	Output	Training	cbow	count
A∘N	**N, P, N**	observed	0.98(1),0.08(5.5),0.13(20.5)	0.82(1),0.17(4.5),0.05(71.5)
A∘N	**N, P, N**	composed	0.99(1),0.02(12), 0.12(24)	0.99(1),0.06(10), 0.05(150.5)

Table 4: Top 1 accuracy (median rank) on the AN→NPN paraphrasing data set. AN phrases are composed and then recursively decomposed into N, (P, N). Comma-delimited scores reported for first noun, preposition, second noun in this order. Training is performed on observed (eq. 2) and composed (eq. 3) phrase representations.

Input	Output	Gold
mountainous region	region in highlands	region with mountains
undersea cable	cable through ocean	cable under sea
underground cavern	cavern through rock	cavern under ground
interdisciplinary field	field into research	field between disciplines
inter-war years	years during 1930s	years between wars
post-operative pain	pain through patient	pain after operation
pre-war days	days after wartime	days before war
intergroup differences	differences between intergroup	differences between minorities
superficial level	level between levels	level on surface

Table 5: Examples of generating NPN phrases from composed ANs.

sions correspond to actual words. As the *cbow* dimensions do not correspond to words, we align the *cbow* spaces by using a small dictionary to learn a linear map which transforms the English vectors into Italian ones as done in Mikolov et al. (2013b). This method (Translation Matrix - TM) is applicable to both *cbow* and *count* spaces. We tune the parameters (TM or SP for *count* and dictionary size 5K or 25K for both spaces) on a standard task of translating English words into Italian. We obtain TM-5K for *cbow* and SP-25K for *count* as optimal settings. The two methods perform similarly for low frequency words while *cbow*-TM-5K significantly outperforms *count*-SP-25K for high frequency words. Our results for the *cbow*-TM-5K setting are similar to those reported by Mikolov et al. (2013b).

Cross-lingual decomposition training Training proceeds as in the monolingual case, this time concatenating the training data sets and estimating a single (de)-composition function for the two languages in the shared semantic space. We train both on observed phrase representations (eq. 2) and on composed phrase representations (eq. 3).

Adjective-noun translation dataset We randomly extract 1,000 AN-AN En-It phrase pairs from a phrase table built from parallel movie subtitles, available at `http://opus.lingfil.uu.se/` (OpenSubtitles2012, en-it) (Tiedemann, 2012).

Input	Output	cbow	count
A∘N(En)	**A,N** (It)	0.31,0.59	0.24,0.54
A∘N (It)	**A,N**(En)	0.50,0.62	0.28,0.48

Table 6: Accuracy of En→It and It→En phrase translation: phrases are composed in source language and decomposed in target language. Training on composed phrase representations (eq. (3)) (with observed phrase training (eq. 2) results are ≈50% lower).

Results are presented in Table 6. While in these preliminary experiments we lack a proper term of comparison, the performance is very good both quantitatively (random < 0.0001) and qualitatively. The En→It examples in Table 7 are representative. In many cases (e.g., *vicious killer*, *rough neighborhood*) we generate translations that are arguably more natural than those in the gold standard. Again, some differences can be explained by different disambiguations (*chest* as *breast*, as in the generated translation, or *box*, as in the gold). Translation into related but not equivalent phrases and generating the same meaning in both constituents (*stellar star*) are again the most significant errors. We also see cases in which this has the desired effect of disambiguating the constituents, such as in the examples in Table 8, showing the nearest neighbours when translating *black tie* and *indissoluble tie*.

Input	Output	Gold
vicious killer	assassino feroce (ferocious killer)	killer pericoloso
spectacular woman	donna affascinante (fascinating woman)	donna eccezionale
huge chest	petto grande (big chest)	scrigno immenso
rough neighborhood	zona malfamata (ill-repute zone)	quartiere difficile
mortal sin	peccato eterno (eternal sin)	pecato mortale
canine star	stella stellare (stellar star)	star canina

Table 7: En→It translation examples (back-translations of generated phrases in parenthesis).

black tie	
cravatta (tie)	**nero** (black)
velluto (velvet)	bianco (white)
giacca (jacket)	giallo (yellow)
indissoluble tie	
alleanza (alliance)	**indissolubile** (indissoluble)
legame (bond)	sacramentale (sacramental)
amicizia (friendship)	inscindibile (inseparable)

Table 8: Top 3 translations of *black tie* and *indissoluble tie*, showing correct disambiguation of *tie*.

	En→It		It→En	
Thr.	Accuracy	Cov.	Accuracy	Cov.
0.00	0.21	100%	0.32	100%
0.55	0.25	70%	0.40	63%
0.60	0.31	32%	0.45	37%
0.65	0.45	9%	0.52	16%

Table 9: AN-AN translation accuracy (*both* A and N correct) when imposing a confidence threshold (random: $1/20K^2$).

7 Generation confidence and generation quality

In Section 3.2 we have defined a search function s returning a list of lexical nearest neighbours for a constituent vector produced by decomposition. Together with the neighbours, this function can naturally return their similarity score (in our case, the cosine). We call the score associated to the top neighbour the *generation confidence*: if this score is low, the vector has no good match in the lexicon. We observe significant Spearman correlations between the generation confidence of a constituent and its quality (e.g., accuracy, inverse rank) in all the experiments. For example, for the AN(En)→AN(It) experiment, the correlations between the confidence scores and the inverse ranks for As and Ns, for both *cbow* and *count* vectors, range between 0.34 ($p < 1e^{-28}$) and 0.42. In the translation experiments, we can use this to automatically determine a subset on which we can translate with very high accuracy. Table 9 shows AN-AN accuracies and coverage when translating only if confidence is above a certain threshold.

Throughout this paper we have assumed that the syntactic structure of the phrase to be generated is given. In future work we will exploit the correlation between confidence and quality for the purpose of eliminating this assumption. As a concrete example, we can use confidence scores to distinguish the two subsets of the AN-NPN dataset introduced in Section 5: the ANs which are paraphrasable with an NPN from those that do not have this property. We assign an AN to the NPN-paraphrasable class if the mean confidence of the PN expansion in its attempted N(PN) decomposition is above a certain threshold. We plot the ROC curve in Figure 1. We obtain a significant AUC of 0.71.

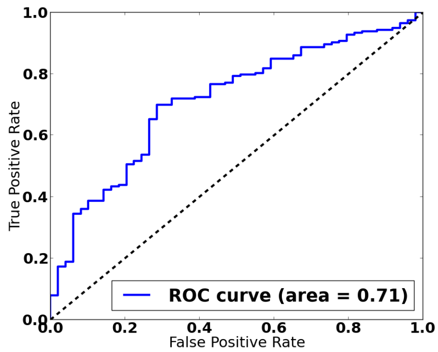

Figure 1: ROC of distinguishing ANs paraphrasable as NPNs from non-paraphrasable ones.

8 Conclusion

In this paper we have outlined a framework for the task of generation with distributional semantic models. We proposed a simple but effective approach to reverting the composition process to obtain meaningful reformulations of phrases through a synthesis-generation process.

For future work we would like to experiment with more complex models for (de-)composition in order to improve the performance on the tasks we used in this paper. Following this, we would like to extend the framework to handle arbitrary phrases, including making (confidence-based) choices on the syntactic structure of the phrase to be generated, which we have assumed to be given throughout this paper.

In terms of applications, we believe that the line of research in machine translation that is currently focusing on replacing parallel resources with large amounts of monolingual text provides an interesting setup to test our methods. For example, Klementiev et al. (2012) reconstruct phrase tables based on phrase similarity scores in semantic space. However, they resort to scoring phrase pairs extracted from an aligned parallel corpus, as they do not have a method to freely *generate* these. Similarly, in the recent work on common vector spaces for the representation of images and text, the current emphasis is on retrieving existing captions (Socher et al., 2014) and not actual generation of image descriptions.

From a more theoretical point of view, our work fills an important gap in distributional semantics, making it a bidirectional theory of the connection between language and meaning. We can now translate linguistic strings into vector "thoughts", and the latter into their most appropriate linguistic expression. Several neuroscientific studies suggest that thoughts are represented in the brain by patterns of activation over broad neural areas, and vectors are a natural way to encode such patterns (Haxby et al., 2001; Huth et al., 2012). Some research has already established a connection between neural and distributional semantic vector spaces (Mitchell et al., 2008; Murphy et al., 2012). Generation might be the missing link to powerful computational models that take the neural footprint of a thought as input and produce its linguistic expression.

Acknowledgments

We thank Kevin Knight, Andrew Anderson, Roberto Zamparelli, Angeliki Lazaridou, Nghia The Pham, Germán Kruszewski and Peter Turney for helpful discussions and the anonymous reviewers for their useful comments. We acknowledge the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January.

Jacob Andreas and Zoubin Ghahramani. 2013. A generative model of vector space semantics. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 91–99, Sofia, Bulgaria.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, To appear, Baltimore, MD.

Marco Baroni. 2013. Composition in distributional semantics. *Language and Linguistics Compass*, 7(10):511–522.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012. Distributional semantics in Technicolor. In *Proceedings of ACL*, pages 136–145, Jeju Island, Korea.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Andrea Frome, Greg Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A deep visual-semantic embedding model. In *Proceedings of NIPS*, pages 2121–2129, Lake Tahoe, Nevada.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala, Sweden.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, pages 771–779, Columbus, OH, USA, June.

James Haxby, Ida Gobbini, Maura Furey, Alumit Ishai, Jennifer Schouten, and Pietro Pietrini. 2001. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293:2425–2430.

Alexander Huth, Shinji Nishimoto, An Vu, and Jack Gallant. 2012. A continuous semantic space describes the representation of thousands of object and action categories across the human brain. *Neuron*, 76(6):1210–1224.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, October. Association for Computational Linguistics.

Colin Kelly, Barry Devereux, and Anna Korhonen. 2012. Semi-supervised learning for automatic conceptual property extraction. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics*, pages 11–20, Montreal, Canada.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of EACL*, pages 130–140, Avignon, France.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *In Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, pages 9–16, Philadelphia, PA, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. `http://arxiv.org/abs/1301.3781/`.

Tomas Mikolov, Quoc Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for Machine Translation. `http://arxiv.org/abs/1309.4168`.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Tom Mitchell, Svetlana Shinkareva, Andrew Carlson, Kai-Min Chang, Vincente Malave, Robert Mason, and Marcel Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320:1191–1195.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Selecting corpus-semantic models for neurolinguistic decoding. In *Proceedings of *SEM*, pages 114–123, Montreal, Canada.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 519–526. Association for Computational Linguistics.

Richard Socher, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809, Granada, Spain.

Richard Socher, Milind Ganjoo, Christopher Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Proceedings of NIPS*, pages 935–943, Lake Tahoe, Nevada.

Richard Socher, Quoc Le, Christopher Manning, and Andrew Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*. In press.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Fabio Zanzotto, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.