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
this paper, we introduce a more direct approach to
phrase generation, inspired by the work in com-
positional 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 pro-
duce 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 dis-
tributed representations of sentences to guide gen-
eration in a machine translation setting. More pre-
cisely, 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 distri-
butional vectors. However, their emphasis is on
reversing the generative story in order to derive
composed meaning representations from word se-
quences. 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. How-
ever, we present a clearer decoupling of synthesis
and generation and we use different (and simpler)
training methods and objective functions. More-
over, Socher and colleagues do not train separate
decomposition rules for different syntactic config-
urations, so it is not clear how they would be able
to control the generation of different output struc-
tures. Finally, the potential for generation is only
addressed in passing, by presenting a few cases
where the generated sequence has the same syn-
tactic structure of the input sequence.

3 General framework

We start by presenting the familiar synthesis set-
ting, focusing on two-word phrases. We then in-
troduce generation for the same structures. Fi-
nally, 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 lex-
icon look-up step to retrieve vectors associated to
words and vice versa: We ignore it in the exposi-
tion 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 \( \vec{p} \):

\[
f_{\text{comp}}(\vec{u}, \vec{v})
\]

where \( \vec{u} \) and \( \vec{v} \) are the vector representations asso-
ciated to words \( u \) and \( v \), \( f_{\text{comp}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \)
(for \( d \) the dimensionality of vectors) is a com-
position function specific to the syntactic relation \( R \)
holding between the two words.\(^1\)

Although we are not bound to a specific com-
position model, throughout this paper we use the
method proposed by Guevara (2010) and Zanzotto
et al. (2010) which defines composition as appli-
cation of linear transformations to the two con-
stituents followed by summing the resulting vec-
tors: \( f_{\text{comp}}(\vec{u}, \vec{v}) = W_1 \vec{u} + W_2 \vec{v} \).
We will further use the following equivalent formulation:

\[
f_{\text{comp}}(\vec{u}, \vec{v}) = W_R[\vec{u}; \vec{v}]
\]

where \( W_R \in \mathbb{R}^{d \times 2d} \) and \( [\vec{u}; \vec{v}] \) is the vertical con-
catenation of the two vectors (using Matlab no-
tation). Following Guevara, we learn \( W_R \) using
examples of word and phrase vectors directly ex-
tracted from the corpus (for the rest of the pa-
per, 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 \( \langle \text{red}, \text{car}, \text{red.car} \rangle, \langle \text{evil}, \text{cat}, \text{evil.cat} \rangle \),
etc. Given a set of training examples stacked into
matrices \( U, V \) (the constituent vectors) and \( P \) (the
the corresponding observed vectors), we estimate \( W_R \)
by solving the least-squares regression problem:

\(^1\)Here we make the simplifying assumption that all vec-
tors have the same dimensionality, however this need not
ecessarily be the case.
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:

$$[\vec{u}; \vec{v}] = f_{\text{decomp}}(\vec{p})$$

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

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:

$$f_{\text{decomp}}(\vec{p}) = W_R' \vec{p}$$

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

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

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

$$\min_{W_R' \in \mathbb{R}^{2d \times d}} \| [U; V] - W_R^* W_R [U; V] \|$$  \hspace{1cm} (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 $s$:

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

where $\text{NN}_\vec{u}$ is a list containing the $t$ nearest neighbours 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: $f_{\text{compAN}}(\vec{big}, f_{\text{compAN}}(\vec{red}, \vec{car}))$, where $f_{\text{compAN}}$ is the composition function for adjective-noun phrase combinations. Conversely, for generation, we decompose the phrase vector with $f_{\text{decompAN}}$. 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 word-form + 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 \((u, v, p)\) tuples for each phrase type, for example, \((\text{red, car, red.car})\) or \((\text{in, car, in.car})\).\(^7\) We concatenate \(u\) and \(v\) vectors to obtain the \([U; V]\) matrix and we use the observed \(\tilde{v}\) 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 \(f_\text{comp}_\text{AN}(\tilde{u}, \tilde{v})\), where \(f_\text{comp}_\text{AN}\) 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 \(f_\text{comp}_\text{AN}(\text{red, car})\)). 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., \(\text{false belief/fallacy, personal appeal/charisma}\)). 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 \(\text{count}\) vectors results are similar to those reported by Dinu and colleagues for the same model, and with \(\text{cbow}\) 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, \(\text{in.the.car}\) and \(\text{in.car}\) occurrences.
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.

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.

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 20K² 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
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 (decomp). We generate a noun from the head slot vector and the “adjective” vector is further decomposed using decomp (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 underwater 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-

---

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

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

8This dataset is available at [http://clic.cimec.unitn.it/composes](http://clic.cimec.unitn.it/composes)
<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Training</th>
<th>cbow</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A◦N N, P, N</td>
<td>N, P, N</td>
<td>observed</td>
<td>0.98(1), 0.08(5.5), 0.13(20.5)</td>
<td>0.82(1), 0.17(4.5), 0.05(71.5)</td>
</tr>
<tr>
<td>A◦N N, P, N</td>
<td>composed</td>
<td>0.99(1), 0.02(12), 0.12(24)</td>
<td>0.99(1), 0.06(10), 0.05(150.5)</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

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

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).

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>cbow</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=N(En)</td>
<td>A,N (It)</td>
<td>0.31</td>
<td>0.59</td>
</tr>
<tr>
<td>A=N (It)</td>
<td>A,N(En)</td>
<td>0.50</td>
<td>0.62</td>
</tr>
</tbody>
</table>

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).

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>vicious killer</td>
<td>assassinoferoce (ferocious killer)</td>
<td>killer pericoloso</td>
</tr>
<tr>
<td>spectacular woman</td>
<td>donna affascinante (fascinating woman)</td>
<td>donna eccezionale</td>
</tr>
<tr>
<td>huge chest</td>
<td>petto grande (big chest)</td>
<td>scrgno immenso</td>
</tr>
<tr>
<td>rough neighborhood</td>
<td>zona malfamata (ill-repute zone)</td>
<td>quareti difficile</td>
</tr>
<tr>
<td>mortal sin</td>
<td>peccato eterno (eternal sin)</td>
<td>pecato mortale</td>
</tr>
<tr>
<td>canine star</td>
<td>stella stellare (stellar star)</td>
<td>star canina</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>black tie</th>
<th>indissoluble tie</th>
</tr>
</thead>
<tbody>
<tr>
<td>cravatta (tie)</td>
<td>alleanza (alliance)</td>
</tr>
<tr>
<td>nero (black)</td>
<td>indissoluble (indissoluble)</td>
</tr>
<tr>
<td>vellato (velvet)</td>
<td>legame (bond)</td>
</tr>
<tr>
<td>bianco (white)</td>
<td>sacramental (sacramental)</td>
</tr>
<tr>
<td>giacca (jacket)</td>
<td>amicizia (friendship)</td>
</tr>
<tr>
<td>giallo (yellow)</td>
<td>inscindibile (inseparable)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Thr.</th>
<th>Accuracy</th>
<th>Cov.</th>
<th>Accuracy</th>
<th>Cov.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.21</td>
<td>100%</td>
<td>0.32</td>
<td>100%</td>
</tr>
<tr>
<td>0.55</td>
<td>0.25</td>
<td>70%</td>
<td>0.40</td>
<td>63%</td>
</tr>
<tr>
<td>0.60</td>
<td>0.31</td>
<td>32%</td>
<td>0.45</td>
<td>37%</td>
</tr>
<tr>
<td>0.65</td>
<td>0.45</td>
<td>9%</td>
<td>0.52</td>
<td>16%</td>
</tr>
</tbody>
</table>

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

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.

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


