DISSECT - *DIS*tributional *SE*mantics *C*omposition *T*oolkit

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

Abstract

We introduce DISSECT, a toolkit to build and explore computational models of word, phrase and sentence meaning based on the principles of distributional semantics. The toolkit focuses in particular on compositional meaning, and implements a number of composition methods that have been proposed in the literature. Furthermore, DISSECT can be useful to researchers and practitioners who need models of word meaning (without composition) as well, as it supports various methods to construct distributional semantic spaces, assessing similarity and even evaluating against benchmarks, that are independent of the composition infrastructure.

1 Introduction

Distributional methods for meaning similarity are based on the observation that similar words occur in similar contexts and measure similarity based on patterns of word occurrence in large corpora (Clark, 2012; Erk, 2012; Turney and Pantel, 2010). More precisely, they represent words, or any other target linguistic elements, as high-dimensional vectors, where the dimensions represent context features. Semantic relatedness is assessed by comparing vectors, leading, for example, to determine that *car* and *vehicle* are very similar in meaning, since they have similar contextual distributions. Despite the appeal of these methods, modeling words in isolation has limited applications and ideally we want to model semantics *beyond word level* by representing the meaning of phrases or sentences. These combinations are infinite and compositional methods are called for to derive the meaning of a larger construction from the meaning of its parts. For this reason, the question of compositionality within the distributional paradigm has received a lot of attention in recent years and a number of compositional frameworks have been proposed in the distributional semantic literature, see, e.g., Coecke et al. (2010) and Mitchell and Lapata (2010). For example, in such frameworks, the distributional representations of *red* and *car* may be combined, through various operations, in order to obtain a vector for *red car*.

The DISSECT toolkit (`http://clic.cimec.unitn.it/composes/toolkit`) is, to the best of our knowledge, the first to provide an easy-to-use implementation of many compositional methods proposed in the literature. As such, we hope that it will foster further work on compositional distributional semantics, as well as making the relevant techniques easily available to those interested in their many potential applications, e.g., to context-based polysemy resolution, recognizing textual entailment or paraphrase detection. Moreover, the DISSECT tools to construct distributional semantic spaces from raw co-occurrence counts, to measure similarity and to evaluate these spaces might also be of use to researchers who are not interested in the compositional framework. DISSECT is freely available under the GNU General Public License.

2 Building and composing distributional semantic representations

The pipeline from corpora to compositional models of meaning can be roughly summarized as consisting of three stages:[1]

1. Extraction of co-occurrence counts from corpora In this stage, an input corpus is used to extract counts of target elements co-occurring with some contextual features. The target elements can vary from words (for lexical similarity), to pairs of words (e.g., for relation categorization),

[1] See Turney and Pantel (2010) for a technical overview of distributional methods for semantics.

to paths in syntactic trees (for unsupervised paraphrasing). Context features can also vary from shallow window-based collocates to syntactic dependencies.

2. Transformation of the raw counts This stage may involve the application of weighting schemes such as Pointwise Mutual Information, feature selection, dimensionality reduction methods such as Singular Value Decomposition, etc. The goal is to eliminate the biases that typically affect raw counts and to produce vectors which better approximate similarity in meaning.

3. Application of composition functions Once meaningful representations have been constructed for the atomic target elements of interest (typically, words), various methods, such as vector addition or multiplication, can be used for combining them to derive context-sensitive representations or for constructing representations for larger phrases or even entire sentences.

DISSECT can be used for the second and third stages of this pipeline, as well as to measure similarity among the resulting word or phrase vectors. The first step is highly language-, task- and corpus-annotation-dependent. We do not attempt to implement all the corpus pre-processing and co-occurrence extraction routines that it would require to be of general use, and expect instead as input a matrix of raw target-context co-occurrence counts.[2] DISSECT provides various methods to re-weight the counts with association measures, dimensionality reduction methods as well as the composition functions proposed by Mitchell and Lapata (2010) (*Additive*, *Multiplicative* and *Dilation*), Baroni and Zamparelli (2010)/Coecke et al. (2010) (*Lexfunc*) and Guevara (2010)/Zanzotto et al. (2010) (*Fulladd*). In DISSECT we define and implement these in a unified framework and in a computationally efficient manner. The focus of DISSECT is to provide an intuitive interface for researchers and to allow easy extension by adding other composition methods.

3 DISSECT overview

DISSECT is written in Python. We provide many standard functionalities through a set of powerful command-line tools, however users with basic Python familiarity are encouraged to use the Python interface that DISSECT provides. This section focuses on this interface (see the online documentation on how to perform the same operations with the command-line tools), that consists of the following top-level packages:

```
#DISSECT packages
composes.matrix
composes.semantic_space
composes.transformation
composes.similarity
composes.composition
composes.utils
```

Semantic spaces and transformations The concept of a semantic space (`composes.semantic_space`) is at the core of the DISSECT toolkit. A semantic space consists of co-occurrence values, stored as a matrix, together with strings associated to the rows of this matrix (by design, the target linguistic elements) and a (potentially empty) list of strings associated to the columns (the context features). A number of transformations (`composes.transformation`) can be applied to semantic spaces. We implement weighting schemes such as positive Pointwise Mutual Information (*ppmi*) and Local Mutual Information, feature selection methods, dimensionality reduction (Singular Value Decomposition (*SVD*) and Nonnegative Matrix Factorization (*NMF*)), and new methods can be easily added.[3] Going from raw counts to a transformed space is accomplished in just a few lines of code (Figure 1).

```
#create a semantic space from counts in
#dense format("dm"): word freq1 freq2 ..
ss = Space.build(data="counts.txt",
                 format="dm")

#apply transformations
ss = ss.apply(PpmiWeighting())
ss = ss.apply(Svd(300))

#retrieve the vector of a target element
print ss.get_row("car")
```

Figure 1: Creating a semantic space.

[2]These counts can be read from a text file containing two strings (the target and context items) and a number (the corresponding count) on each line (e.g., `maggot food 15`) or from a matrix in format `word freq1 freq2 ...`

[3]The complete list of transformations currently supported can be found at `http://clic.cimec.unitn.it/composes/toolkit/spacetrans.html#spacetrans`.

```
#load a previously saved space
ss = io_utils.load("ss.pkl")

#compute cosine similarity
print ss.get_sim("car", "book",
                 CosSimilarity())

#the two nearest neighbours of "car"
print ss.get_neighbours("car", 2,
                        CosSimilarity())
```

Figure 2: Similarity queries in a semantic space.

Furthermore DISSECT allows the possibility of adding new data to a semantic space in an online manner (using the `semantic_space.peripheral_space` functionality). This can be used as a way to efficiently expand a co-occurrence matrix with new rows, without re-applying the transformations to the entire space. In some other cases, the user may want to represent phrases that are specialization of words already existing in the space (e.g., *slimy maggot* and *maggot*), without distorting the computation of association measures by counting the same context twice. In this case, adding *slimy maggot* as a "peripheral" row to a semantic space that already contains *maggot* implements the desired behaviour.

Similarity queries Semantic spaces are used for the computation of similarity scores. DISSECT provides a series of similarity measures such as cosine, inverse Euclidean distance and Lin similarity, implemented in the `composes.similarity` package. Similarity of two elements can be computed within one semantic space or across two spaces that have the same dimensionality. Figure 2 exemplifies (word) similarity computations with DISSECT.

Composition functions Composition functions in DISSECT (`composes.composition`) take as arguments a list of element pairs to be composed, and one or two spaces where the elements to be composed are represented. They return a semantic space containing the distributional representations of the composed items, which can be further transformed, used for similarity queries, or used as inputs to another round of composition, thus scaling up beyond binary composition. Figure 3 shows a Multiplicative composition example. See Table 1 for the currently available composition models, their definitions and parameters.

Model	Composition function	Parameters
Add.	$w_1\vec{u} + w_2\vec{v}$	$w_1(=1), w_2(=1)$
Mult.	$\vec{u} \odot \vec{v}$	-
Dilation	$\|\|\vec{u}\|\|_2^2\vec{v} + (\lambda - 1)\langle\vec{u}, \vec{v}\rangle\vec{u}$	$\lambda(=2)$
Fulladd	$W_1\vec{u} + W_2\vec{v}$	$W_1, W_2 \in \mathbf{R}^{m\times m}$
Lexfunc	$A_u\vec{v}$	$A_u \in \mathbf{R}^{m\times m}$

Table 1: Currently implemented composition functions of inputs (u, v) together with parameters and their default values in parenthesis, where defined. Note that in Lexfunc each functor word corresponds to a separate matrix or tensor A_u (Baroni and Zamparelli, 2010).

Parameter estimation All composition models except Multiplicative have parameters to be estimated. For simple models with few parameters, such as as Additive, the parameters can be passed by hand. However, DISSECT supports automated parameter estimation from training examples. In particular, we extend to all composition methods the idea originally proposed by Baroni and Zamparelli (2010) for Lexfunc and Guevara (2010) for Fulladd, namely to use corpus-extracted example vectors of both the input (typically, words) and output elements (typically, phrases) in order to optimize the composition operation parameters. The problem can be generally stated as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} ||P - fcomp_{\boldsymbol{\theta}}(U, V)||_F$$

where U, V and P are matrices containing input and output vectors respectively. For example U may contain adjective vectors such as *red*, *blue*, V noun vectors such as *car*, *sky* and P corpus-extracted vectors for the corresponding phrases *red car*, *blue sky*. $fcomp_{\boldsymbol{\theta}}$ is a composition function and $\boldsymbol{\theta}$ stands for a list of parameters that this composition function is associated with.[4] We implement standard least-squares estimation methods as well as Ridge regression with the option for generalized cross-validation, but other methods such as partial least-squares regression can be easily added. Figure 4 exemplifies the Fulladd model.

Composition output examples DISSECT provides functions to evaluate (compositional) distributional semantic spaces against benchmarks in the `composes.utils` package. However, as a more qualitatively interesting example of what can be done with DISSECT, Table 2 shows the nearest

[4]Details on the extended corpus-extracted vector estimation method in DISSECT can be found in Dinu et al. (2013).

```
#instantiate a multiplicative model
mult_model = Multiplicative()

#use the model to compose words from input space input_space
comp_space = mult_model.compose([("red", "book", "my_red_book"),
                                 ("red", "car", "my_red_car")],
                                 input_space)

#compute similarity of: 1) two composed phrases and 2) a composed phrase and a word
print comp_space.get_sim("my_red_book", "my_red_car", CosSimilarity())
print comp_space.get_sim("my_red_book", "book", CosSimilarity(), input_space)
```

Figure 3: Creating and using Multiplicative phrase vectors.

```
#training data for learning an adjective-noun phrase model
train_data = [("red","book","red_book"), ("blue","car","blue_car")]

#train a fulladd model
fa_model = FullAdditive()
fa_model.train(train_data, input_space, phrase_space)

#use the model to compose a phrase from new words and retrieve its nearest neighb.
comp_space = fa_model.compose([("yellow", "table", "my_yellow_table")], input_space)
print comp_space.get_neighbours("my_yellow_table", 10, CosSimilarity())
```

Figure 4: Estimating a Fulladd model and using it to create phrase vectors.

Target	**Method**	**Neighbours**
florist	Corpus	Harrod, wholesaler, stockist
flora + -ist	Fulladd	flora, fauna, ecologist
	Lexfunc	ornithologist, naturalist, botanist
	Additive	flora, fauna, ecosystem

Table 3: Compositional models for morphology. Top 3 neighbours of *florist* using its (low-frequency) corpus-extracted vector, and when the vector is obtained through composition of *flora* and *-ist* with Fulladd, Lexfunc and Additive.

neighbours of *false belief* obtained through composition with the Fulladd, Lexfunc and Additive models. In Table 3, we exemplify a less typical application of compositional models to derivational morphology, namely obtaining a representation of *florist* compositionally from distributional representations of *flora* and *-ist* (Lazaridou et al., 2013).

4 Main features

Support for dense and sparse representations Co-occurrence matrices, as extracted from text, tend to be very sparse structures, especially when using detailed context features which include syntactic information, for example. On the other hand, dimensionality reduction operations, which are often used in distributional models, lead to smaller, dense structures, for which sparse representations are not optimal. This is our motivation for supporting both dense and sparse representations. The choice of dense vs. sparse is initially determined by the input format, if a space is created from co-occurrence counts. By default, DISSECT switches to dense representations after dimensionality reduction, however the user can freely switch from one representation to the other, in order to optimize computations. For this purpose DISSECT provides wrappers around matrix operations, as well as around common linear algebra operations, in the `composes.matrix` package. The underlying Python functionality is provided by `numpy.array` and `scipy.sparse`.

Efficient computations DISSECT is optimized for speed since most operations are cast as matrix operations, that are very efficiently implemented in Python's `numpy` and `scipy` modules[5]. Tables 4 and 5 show running times for typical DISSECT operations: application of the *ppmi* weighting scheme, nearest neighbour queries and estimation of composition function parameters (on a 2.1

[5]For SVD on sparse structures, we use `sparsesvd` (`https://pypi.python.org/pypi/sparsesvd/`). For NMF, we adapted `http://www.csie.ntu.edu.tw/~cjlin/nmf/` (Lin, 2007).

Target	Method	Neighbours
belief	Corpus	moral, dogma, worldview, religion, world-view, morality, theism, tenet, agnosticism, dogmatic
false belief	Fulladd	pantheist, belief, agnosticism, religiosity, dogmatism, pantheism, theist, fatalism, deism, mind-set
	Lexfunc	self-deception, untruth, credulity, obfuscation, misapprehension, deceiver, disservice, falsehood
	Additive	belief, assertion, falsity, falsehood, truth, credence, dogma, supposition, hearsay, denial

Table 2: Top nearest neighbours of *belief* and of *false belief* obtained through composition with the Fulladd, Lexfunc and Additive models.

Method	Fulladd	Lexfunc	Add.	Dilation
Time (s.)	2864	787	46	68

Table 4: Composition model parameter estimation times (in seconds) for 1 million training points in 300-dimensional space.

Matrix size (nnz)	Ppmi	Query
100Kx300 (30M)	5.8	0.5
100Kx100K (250M)	52.6	9.5

Table 5: Running times (in seconds) for 1) application of *ppmi* weighting and 2) querying for the top neighbours of a word (cosine similarity) for different matrix sizes (nnz: number of non-zero entries, in millions).

GHz machine). The price to pay for fast computations is that data must be stored in main memory. We do not think that this is a major inconvenience. For example, a typical symmetric co-occurrence matrix extracted from a corpus of several billion words, defining context in terms of 5-word windows and considering the top 100K$\times$100K most frequent words, contains $\approx$ 250 million entries and requires only 2GB of memory for (double precision) storage.

Simple design We have opted for a very simple and intuitive design as the classes interact in very natural ways: A semantic space stores the actual data matrix and structures to index its rows and columns, and supports similarity queries and transformations. Transformations take one semantic space as input to return another, transformed, space. Composition functions take one or more input spaces and yield a composed-elements space, which can further undergo transformations and be used for similarity queries. In fact, DISSECT semantic spaces also support higher-order tensor representations, not just vectors. Higher-order representations are used, for example, to represent transitive verbs and other multi-argument functors by Coecke et al. (2010) and Grefenstette et al. (2013). See `http://clic.cimec.unitn.it/composes/toolkit/composing.html` for an example of using DISSECT for estimating such tensors.

Extensive documentation The DISSECT documentation can be found at `http://clic.cimec.unitn.it/composes/toolkit`. We provide a tutorial which guides the user through the creation of some toy semantic spaces, estimation of the parameters of composition models and similarity computations in semantic spaces. We also provide a full-scale example of intransitive verb-subject composition. We show how to go from co-occurrence counts to composed representations and make the data used in the examples available for download.

Comparison to existing software In terms of design choices, DISSECT most resembles the Gensim toolkit (Řehůřek and Sojka, 2010). However Gensim is intended for topic modeling, and therefore diverges considerably from DISSECT in its functionality. The SSpace package of Jurgens and Stevens (2010) also overlaps to some degree with DISSECT in terms of its intended use, however, like Gensim, it does not support compositional operations that, as far as we know, are an unique feature of DISSECT.

5 Future extensions

We implemented and are currently testing DISSECT functions supporting other composition methods, including the one proposed by Socher et al. (2012). Adding further methods is our top-priority goal. In particular, several distributional models of word meaning in context share important similarities with composition models, and we plan to add them to DISSECT. Dinu et al. (2012) show, for example, that well-performing, simplified variants of the method in Thater et al. (2010), Thater et al. (2011) and Erk and Padó (2008) can be reduced to relatively simple matrix operations, making them particularly suitable for a DISSECT implementation.

DISSECT is currently optimized for the composition of many phrases of the same type. This is in line with most of the current evaluations of compositional models, which focus on specific phenomena, such as adjectival modification, noun-noun compounds or intransitive verbs, to name a few. In the future we plan to provide a module for composing entire sentences, taking syntactic trees as input and returning composed representations for each node in the input trees.

Finally, we intend to make use of the existing Python plotting libraries to add a visualization module to DISSECT.

6 Acknowledgments

We thank Angeliki Lazaridou for helpful discussions. This research was supported by the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

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.

Stephen Clark. 2012. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics, 2nd edition*. Blackwell, Malden, MA. In press.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Georgiana Dinu, Stefan Thater, and Sören Laue. 2012. A comparison of models of word meaning in context. In *Proceedings of NAACL HLT*, pages 611–615, Montreal, Canada.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. A general framework for the estimation of distributional composition functions. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria. In press.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP*, pages 897–906, Honolulu, HI.

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

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of IWCS*, pages 131–142, Potsdam, Germany.

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

David Jurgens and Keith Stevens. 2010. The S-Space package: an open source package for word space models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 30–35, Uppsala, Sweden.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of ACL*, Sofia, Bulgaria. In press.

Chih-Jen Lin. 2007. Projected gradient methods for Nonnegative Matrix Factorization. *Neural Computation*, 19(10):2756–2779.

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

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of ACL*, pages 948–957, Uppsala, Sweden.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of IJCNLP*, pages 1134–1143, Chiang Mai, Thailand.

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

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.