A Vector Model for Type-Theoretical Semantics

Konstantin Sokolov
Peter the Great Polytechnic University, St. Petersburg, Russia
sokolov@dcn.icc.spbstu.ru

Abstract

Vector models of distributional semantics can be viewed as a geometric interpretation of a fragment of dependent type theory. By extending to a bigger fragment to include the dependent product we achieve a significant increase in expressive power of vector models, which allows for an implementation of contextual adaptation of word meanings in the compositional setting.

1 Introduction

In this paper we discuss a possibility of reconciling two distinct threads of research in computational lexical semantics, namely distributional semantics (Lenci, 2008; Baroni et al., 2014) and formal semantics based on dependent type theory (Ranta, 1994; Luo, 2012). Although both approaches focus on related problems of lexical semantics and composition, it is hard to combine them in a unified framework because of differences in computational techniques they employ. However, their theoretical foundations are compatible and it is possible to incorporate vector space models into a computational framework based on dependent type theory. The extensions of existing vector representations required for this task are motivated by the geometric interpretation of type theory.

The reason why such extensions are necessary is the limitations of the function application model of compositionality. In formal semantics compositionality is traditionally modeled as an application of one logical form to another, technically realized as a reduction of concatenated logical forms expressed in a variant of typed lambda calculus (Montague, 1975). In distributional semantics this approach is retained, although lambda terms are substituted with vectors and tensors of various ranks (Baroni et al., 2014). This model of compositionality has difficulties with the compositional treatment of certain common types of natural language expressions, that require context-dependence of terms and a non-trivial machanism of meaning adaptation. Paradigmatic examples of such linguistic phenomena are logical polysemy and co-predication, which received formal treatment in the works of Pustejovsky (1995) and are among the central problems of type-theoretical semantics (Cooper, 2005; Asher, 2011; Luo, 2012).

The problems related to logical polysemy include cases where an intended aspect of meaning of an argument requires an adaptation of a predicate which is applied to it and vice versa. Consider the case of an adjective applied to a noun in *red apple*, *red watermelon* and *red crystal* (Lahav, 1989). In these examples a notion of “redness” is different for each of the arguments. A red apple is red when most of its surface is red, a watermelon is red inside, a crystal is red entirely. It is possible to characterize a watermelon as being ripe by saying that it is red, which is not the case for a crystal. As a result, an adaptation of a predicate is needed to properly account for the meaning variations in all these cases. Similarly, an argument can have multiple aspects of meaning selected by various predicates. In *heavy book* and *boring book* a book is treated as either a physical object or an information content. Both aspects clearly correspond to different sets of properties, which are taken into account by the predicates and reflect in the systemic organization of the lexicon, e. g. hypernym relations between synsets or relations between word classes (Hanks, 1996). Words both in argument and predicate positions can demonstrate capability of contextual meaning adaptation, sometimes even simultaneously, like in *red book*. Such cases violate the direction of application of a predicate and are problematic for the function application model of composition.

Another violation occurs in cases of co-predication, which takes place when two or more predicates make use of conflicting meaning aspects of an argument, leading to the necessity to assign at least two incompatible meaning representations (e. g. types) to a single word. The "lunch sentence" proposed by Pustejovsky (1995) is a classic example:

A lunch was delicious but took forever.

Notwithstanding these obstacles, we find that the function application model of composition can in fact be made compatible with context-dependent meaning alternations. An extension required for this task follows naturally from an identification of basic operations of composition in vector models and computation rules in type theory. The resulting model can in turn be given an intuitive geometric interpretation, that greatly clarifies the options for its computational implementation. Recent developments in type theory made clear a tight connection between computation and geometry (Univalent Foundations Program, 2013). In the following sections we give a sketch of a method of incorporating semantic vector spaces within a type-theoretical framework of Luo by interpreting primitive types as vector spaces, dependent types as fibers of a vector bundle and so forth. Currently we limit ourselves to the dependent product type, leaving an analysis of the dependent sum and dot types for later. However, we give some general remarks as to how these important constructs might be implemented in our setting.

2 Related Work

In this section we give an overview of the general framework of composition currently adopted in distributional semantic, followed by a brief discussion of type-theoretical semantics.

2.1 Compositional Vector Models

Lack of support for compositionality remains a serious limitation of distributional semantics, although many techniques were proposed to enable compositional treatment of vector representations. These proposals include methods based on vector addition and multiplication (Mitchell and Lapata, 2010), tensor operations (Smolensky, 1990; Widdows, 2008), linear maps (Baroni et al., 2014; Coecke et al., 2010), tensor decomposition (Van de Cruys et al., 2013), co-composition based on vector space projections (Tsubaki et al., 2013), simultaneous processing of meaning and composition in neural embeddings (Socher et al., 2013; Pennington et al., 2014) and more. There is an evident tradeoff between expressiveness and computational properties of the compositional vector models. Simple additive and multiplicative models cannot capture important properties of composition in natural language, such as its non-commutative character, relation to syntax, polysemy and contextual meaning adaptation. Complex models generally make use of tensors of various ranks to represent different word types and linear maps that operate on them and in principle are better suited for compositional analysis. However, the actual implementations are very difficult to train and to date no methods of training such models on a large scale were proposed.

Although commutative (vector mixture) methods of composition are easier to implement, non-commutativity is dictated by both linguistic properties and the properties of vector representations. In the analysis of adjective-noun pairs Baroni and Zamparelli (2010) used an asymmetric function application model of compositionality, where an adjective is represented as a matrix applied to a noun vector to produce a new vector. This approach turned into a large research program of compositional distributional semantics (Baroni et al., 2014), which is currently widely accepted.

A similar program was proposed earlier by Coecke et al. (2010), where category theory was used to devise a method of validating paths of composition accross multiple vector spaces specific for various word types in order to arrive to a distinguished "sentence" space, where comparison of the meanings of sentences could be performed. The composition here is modeled on the basis of a linear map sending a tensor product of meaning representations of words (i. e. their superposition) to a vector in the sentence space. The particular structure of this linear map is determined by the sentence structure explicated as a reduction in the pregroup grammar formalism. Since linear maps $V \to W$ are actually in a bijective correspondence with vectors in a tensor product space $V \otimes W$, the actual procedure of computation of the sentence meaning consists in "carving out" the right sequence of function applications from the space of all possible reduction paths. The model of composition here is slightly different, but it does not diverge too much from Baroni's proposal. Under the aforementioned bijection a transitive verb can also be viewed as a function applied to the meaning representations of its participants to produce a vector in the sentence space.

Contextual variation of meaning in relation to word sense disambiguation and semantic similarity has been investigated almost since inception of the field (Schütze, 1998; Thater et al., 2010; Thater et al., 2011). However, disambiguation is different from composition (Kartsaklis et al., 2013). Attempts to give an analysis of logical polysemy and contextual adaptation in the compositional setting are limited (Erk and Padó, 2008; Tsubaki et al., 2013).

2.2 Type-Theoretical Semantics

Type-theoretical semantics is an approach to modeling semantics of natural language initiated by Ranta (1994) and heavily influenced by Pustejovsky's Generative Lexicon (Pustejovsky, 1991; Pustejovsky, 1995). It proved capable of giving a convincing analysis of logical polysemy, copredication and systemic organization of the lexicon modeled as subtyping. Although it belongs roughly in the tradition of Montague Grammar (Montague, 1975), it diverges from formal semantics in a number of ways. The distinction between a formal meaning representation and its interpretation in a model is not present. Meaning representations are type expressions and their justification is achieved by an effective computational process. As a consequence, type-theoretical meaning representations have a strong connection with computability due to the Curry-Howard correspondence. Unlike in Montague Grammar, in type-theoretical semantics words are not treated as atomic entities. Instead, their meaning is analysed on the basis of argument-predicate structures. The meaning of a predicate is represented as a function type, which incorporates types of its arguments to the effect similar to selectional restrictions in the study of verb classes (Levin, 1993). Explication of a word meaning through argument type restrictions, i. e. constraints on co-occurence, leads to a duality between argument-predicate structure and lexical meaning (Pustejovsky, 2013). To our view, this notion of duality is closely related to the distributional hypothesis of Z. Harris and justifies an attempt to give a unified treatment of lexical semantics based on both theories.

Current proposals are targeted at developing a formal system that would incorporate words as either terms of a certain type or as types themselves and provide a set of type formation, introduction, elimination and computation rules to be used in derivations. Traditional formal semantics is based on the simply typed lambda calculus à la Church and is given a set-theoretical interpretation. A type-theoretical formalism proposed by Asher and Pustejovsky (2006) builds on a type theory which is close to that used in formal semantics. The distinction between terms and types is retained, a subtyping relation is inherited from the Generative Lexicon and is modeled on the basis of a subsumption relation. In (Asher, 2011) a conflict between subsumptive subtyping and an intended interpretation of dot types is resolved by modifications of the interpertation procedure, which is based on a category-theoretic interpretation in a topos. Other systems (Cooper, 2005; Mery et al., 2007; Luo, 2012) are based on Martin-Löf's dependent type theory, where the boundary between terms and types is blurred. The framework of dependent type semantics (Luo, 2012) makes use of a special mechanism of coercive subtyping, which will be characterized in the next section.

An obvious limitation of type-theoretical semantics is that to date no method of large scale building of such representations on the basis of real-world data has been given, which greatly hinders empirical evaluation. A possible way to overcome this limitation is to adapt logical form learning techniques developed for other types of symbolic semantic representation (Zettlemoyer and Collins, 2012; Liang and Potts, 2015).

3 Formal Background

In this section we give a motivation for the use of dependent types in formal semantics, followed by a brief overview of the coercive subtyping framework of Luo as applied to natural language expressions. The notion of a vector bundle plays a crucial role in our interpretation, so we also give here the definition.

3.1 Dependent Type Theory

Martin-Löf's type theory can be viewed as a formal system of deduction. When used as a metalanguage, a type theory can incorporate both rules of formation of logical statements and deduction rules of a logical system. The basic elements of a formal system of type theory are judgements of various forms. A judgement that proposition A is true is written A *true*, $a : A$ is a judgement saying that a is an element of type A, a judgement $a = b : A$ says that a and b are equal objects of type A. Besides simple judgements of the forms given above there are hypothetical judgements that depend on another judgements, e. g. $f(a) : B \ (a : A)$ says that an object $f(a)$ is of type B given that a is an object of type A. With hypothetical judgements and a number of deduction rules which can bind hypotheses it is possible to construct a derivation of a particular judgement, which does not depend on any hypothesis, i. e. a derivation in a system of natural deduction for judgements. Such a system can be presented in two forms, the so called Prawitz and Gentzen style natural deduction, the latter uses sequents, i. e. expressions involving typing contexts and a turnstile.

As a short example, consider the introduction and elimination rules for conjunction implemented inside a metalanguage of type theory (Gentzen style):

$$\frac{\Gamma \vdash A \ \textit{true} \qquad \Gamma \vdash B \ \textit{true}}{\Gamma \vdash A \wedge B \ \textit{true}} \, I_\wedge$$

$$\frac{\Gamma \vdash A \wedge B \ \textit{true}}{\Gamma \vdash A \ \textit{true}} \, E_\wedge \qquad \frac{\Gamma \vdash A \wedge B \ \textit{true}}{\Gamma \vdash B \ \textit{true}} \, E_\wedge$$

Since all components of a logical system are immersed in the same metalanguage, it is possible to extend the language by allowing types to depend on terms of another types, which significantly raises expressivity. A dependent type is written $A(x)$, and two special type expressions are introduced, the dependent product type $\Pi(x : A)B(x)$ and the dependent sum type $\Sigma(x : A)B(x)$. When there is no actual dependency of $B(x)$ on the elements of A, the product type simplifies to a function type $A \rightarrow B$, and the sum type to a direct product $A \times B$.

To establish an equality of two terms of a type it is necessary to reduce each of them to their corresponding canonical objects. Then an equality judgement is justified by the syntactic equivalence of canonical objects. As an example, consider an inductive type for natural numbers $\mathbb{N}$, which has two type constructors $0 : \mathbb{N}$ and $succ(n) : \mathbb{N} \ (n : \mathbb{N})$. To prove $1 + 1 = 2 : \mathbb{N}$ both sides of the equality are reduced to the form $succ(succ(0))$. The rules used for reducing an object to its canonical form are called computation rules, reduction itself is often called computation.

Types and propositions are identified. For instance, in the judgement A *true* a symbol A is treated as a proposition assumed to be true, whereas in $a : A$ it is treated as a type. Under such identification an object of type A is treated as an evidence (a proof object) that proposition A is true. To prove A is to construct an object of type A using the rules of the system. That justifies the possibility to use type theory for formal semantics, since to say that a proposition is provable is to say that its truth conditions are satisfied (Ranta, 1994).

One of the main obstacles to applying dependent type theory to the task of modeling lexical semantics is that the subtyping relation, which naturally represents hypernym relations between concepts and is traditionally modeled as subsumption (Pustejovsky, 1995), is incompatible with the notion of canonical object (Luo et al., 2013). For example, the subsumptive subtyping justifies inferences of the form

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A < B}{\Gamma \vdash a : B}$$

In case of dependent types, an inference like

$$\frac{\Gamma \vdash a : \textit{List}(A) \qquad \Gamma \vdash A < B}{\Gamma \vdash a : \textit{List}(B)}$$

is incorrect. To justify that we would need to reduce both $a : List(A)$ and $a : List(B)$ to the same canonical representation, which is impossible to do since canonical objects of these two types are built with different sets of type constructors.

To solve this Luo (2012) proposed a mechanism of type coercions, which allows for the substitution of a term of a subtype in a context where a term of its supertype is required. The actual coercion is achieved by applying a coercion function to the term. A coercion judgement of the form $\Gamma \vdash A <_c B : \textit{Type}$ states that objects of the form $c(a)$, where a is an object of type A and c is a coercion function, can be used in contexts where objects of type B are expected. By using such a mechanism it becomes possible to justify relations between predicates such as $[\![\textit{human}]\!] \rightarrow [\![\textit{book}]\!] \rightarrow \textit{Prop} < [\![\textit{man}]\!] \rightarrow \Sigma([\![\textit{book}]\!], [\![\textit{heavy}]\!]) \rightarrow \textit{Prop}$ by declaring coercions $[\![\textit{man}]\!] <_{c_1} [\![\textit{human}]\!]$ and $\Sigma(A, B) <_{p_1} A$. The resulting subtyping is contravariant in arguments as is usually expected. Coercive subtyping is a conservative extension (in the weak sense) of dependent type theory, s. (Luo et al., 2013) for details.

3.2 Vector Bundles

Intuitively, a vector bundle is a set of vector spaces parameterized by points of some topological space. Formal definitions follow (Luke and Mishchenko, 2013).

A vector bundle is a continuous map $p : E \to B$ s. t. $p^{-1}(b)$ is a vector space for each $b \in B$. Additionally, there is an open cover $\{U_\alpha\}$ of B and for each U_α there exists a homeomorphism $h_\alpha : p^{-1}(U_\alpha) \to U_\alpha \times \mathbb{R}^k$ s. t. $h_\alpha(p^{-1}(b))$ is a vector space isomorphic to $\{b\} \times \mathbb{R}^k$ for each $b \in U_\alpha$. This is called a local trivialization condition. The resulting construct is a locally trivial real vector bundle of rank k (we say vector bundle for short), E is the total space, B is the base space, $p^{-1}(b)$ is a fiber over b.

A pair of trivializations $h_\alpha : p^{-1}(U_\alpha) \to U_\alpha \times \mathbb{R}^k$ and $h_\beta : p^{-1}(U_\beta) \to U_\beta \times \mathbb{R}^k$ induces a map $h_\alpha h_\beta^{-1} : (U_\alpha \cap U_\beta) \times \mathbb{R}^k \to (U_\alpha \cap U_\beta) \times \mathbb{R}^k$ called transition function. Transitions can be thought of as continuous changes of coordinates.

4 Vector Models for Dependent Types

In this section we discuss a geometric interpretation of a dependent type system, resulting from the treatment of types as vector spaces. Such an interpretation provides insights into a possible computational implementation.

4.1 Geometric Interpretation

The usual way to represent words in vector models is to identify them with vectors. In our approach, instead of using vectors as primitive elements, we switch to an equivalence class of vectors w. r. t. multiplication by a scalar. Such a modification does not affect our ability to compute the cosine similarity between primitive elements. We do not identify equivalence classes of vectors with words right away. Instead, it is more convenient to think of a word as a region or a neighbourhood in the space obtained from the initial vector space by the factorization we have just described, cf. (Erk, 2009). We denote that initial vector space A and treat it as a primitive type of our system. This approach allows to make all types and their objects be vector spaces. For example, if $A = \mathbb{R}^2$, then $a : A$ is a one-dimensional linear subspace of A.

Function types are built recursively with an arrow constructor. The simplest possible function type in our system is $A \to A$, objects of this type are linear operators on A. Analogously to the primitive type, we would like to consider equivalence classes of operators as objects of that type.

A dependent product $\Pi(x : A)B(x)$ is interpreted as a vector bundle. Its fibers are vector spaces, parameterized by points of the base space. An easy way to imagine this situation is to consider an orthogonal complement of a one-dimensional subspace in $\mathbb{R}^3$ with the usual scalar multiplication, which is isomorphic to $\mathbb{R}^2$. Then for any one-dimensional subspace in $\mathbb{R}^2$ there is a corresponding orthogonal subspace of dimension two, which is a fiber of a vector bundle $p : E \to B$. A section of a vector bundle is a map $s : U \to E$, where U is an open subset of B, such that $p(s(b)) = b$. We interpret predicates as global sections, which have B as the domain and send every point $b \in B$ to some vector in the fiber $p^{-1}(b)$.

Consider the previous example, where the base space is a two-dimensional euclidean space embedded in $\mathbb{R}^3$ and the fibers are two-dimensional subspaces orthogonal to the lines in the base space. We can view this construct as a real vector bundle of rank two. A global section sends in a continuous manner a vector from the base space to some vector in the plane orthogonal to it.

We summarize corespondences between variuos interpretations in Table 1.

4.2 Computation

The notion of canonical object is central to the dependent type theory. The computation of a canonical representation of an object of some type is achieved by a series of reductions in an order prescribed by the structure of that type. More precisely, application of an object of type $\Pi(x : A)B(x)$ to an object of type A amounts to selecting a point from the "result space" $B(a)$ parameterized by the argument, given a point $a : A$ as an input:

$$\frac{\Gamma \vdash f : \Pi(x : A)B(x) \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathit{app}(f, a, \Pi(x : A)B(x), A) = f(a) : B(a)} \, C_\Pi$$

Elements of the same type are comparable, whereas elements of different types are not. For instance, objects of type $B(x)$, which is the type of a fiber, are not comparable with objects of type A, which is the type of the base. This is exactly

Type theory	Vector model	Geometric interpretation	Linguistic interpretation
element $x\!:\!A$	a vector in $\mathbb{R}^2$	a point in the base space B	a word
dependent type $A(x)$	a vector space parameterized by $x \in \mathbb{R}^2$	a fiber $p^{-1}(x)$	contextual modifications of words w. r. t a word x
product type $\Pi(x\!:\!A)B(x)$	linear maps parameterized by vectors in $\mathbb{R}^2$	a vector bundle	a set of predicates adapted to an argument
element $f\!:\!\Pi(x\!:\!A)B(x)$	a parameterized linear map	a global section	an adapted predicate

Table 1: Correspondences between interpretations.

where the coercive subtyping shows up. Recall that coercions are formulated in such a way as to make it possible to use an object of some type instead of an object of its supertype in a given context. Also note, that in the framework of Luo coercions are maps. To make things easier, currently we impose a restriction on vector bundles that the dimensionality of fibers be equal to the dimensionality of the base space. Note, that the general definition of a vector bundle does not require that. Then coersions from $B(x)$ to A can be implemented uniformly as trivial maps from fibers to the base, which we normally omit writing down explicitly. Comparing objects of $B(a)$ to objects of $B(b)$, given that a and b are points in the base space, does not require any special arrangements. Coercions in that case are determined by the usual transition functions.

4.3 Implementation

We give an example of the construct for the simplest possible case.

Let $A \approx \mathbb{R}^2$ be a distinguished plane of a three-dimensional euclidean space. Its one-dimensional subspaces are equivalence classes of vectors with respect to multiplication by a scalar. We build a vector bundle $p : E \to A$ with a fiber isomorphic to $\mathbb{R}^2$ and therefore also isomorphic to the base space. It is natural to use an associated projective space instead of $\mathbb{R}^2$, denoted $P(\mathbb{R}^2)$. Projectivization has a number of advantages. Under identification of projective points with rotations of an underlying $\mathbb{R}^2$, it can be seen as a compact Lie group, namely a projective special orthogonal group $PSO(2, \mathbb{R})$. The usual cosine similarity measure for words translates to an additive metric on $P(\mathbb{R}^2)$. In the induced topology the notion of a neighborhood of a word is well-defined. Analogously, switching to an associated projective bundle $P(E)$ allows us to treat fibers in the same manner. As usual, we can represent $P(\mathbb{R}^2)$ as a circle S^1 with opposite points identified, then the projective bundle can be represented as a torus $S^1 \times S^1$, this time with four antipodal points identified. Care is needed to correctly assign orientations on the fibers.

Sections can be viewed as vector-valued functions defined on the base space. Since in our case the vectors can be obtained by an action of $PSO(2, \mathbb{R})$, a single parameter is sufficient to encode the vectors. This parameter is actually an angle between the zero and the value vectors in the fiber. That allows us to represent each section as a big cirle on a torus and to encode the required value as an angular offset assigned to each point. Such a scalar function must be smooth and periodic to satisfy the properties of a vector bundle, and it suffices to limit its range to $[0, \pi)$, since we sum angles modulo π. The function can be approximated with a square table with entries periodic accross both rows and columns. It is easy to vizualize the scalar function on a torus as a heat map or as a 2D surface over the square, with the height of a point being equal to the value of the function at that point. The periodicity and smoothness requirements suggest that it should be possible to approximate that surface with a partial weighted sum of two-dimesional harmonics.

5 Discussion

A characteristic trait of our model is that both arguments and predicates are treated as entities of a continuous nature. Their co-adaptation turns into a process of evaluation of stability areas where small changes of both the predicate and the argument do not lead to drastic changes in the meaning of a composite expression. This is a property that we pursue by intention and it is reminiscent of multiple examples of zonal reasoning in cognitive linguistics and perception (Stevens, 1972; Gärdenfors, 2004). Symbolic representations are often considered to be incompatible with that type of meaning representations. However, a more expressive formalism like that of dependent type theory makes the boundary less apparent. It is still to be determined whether our model can be given a sound interpretation in terms of cognitive lingusitics.

Early vector models were based on words cooccurence in a corpus and required very high dimensionality of representations (Deerwester et al., 1990; Landauer and Dumais, 1997). Vector coordinates could be interpreted as contextual cooccurence counts or as weights of the latent factors, depending on the model. Later the field was revolutionized by introduction of machine learning techniques, which allowed to approximate low dimensional vector representations (Mikolov et al., 2013; Pennington et al., 2014). It made the problem more tractable, while at the same time it became impossible to interpret single coordinates, as in these approaches the size of the vectors is determined on the basis of the desired accuracy of approximation and not the actual counts. In our model we consider the primitive elements of representation as abstract vectors. We also consider complex mathematical structures such as inverse image and fibration as parts of our representation. As a consequence, it would be incorrect to compare the dimensionality of a local trivialization of a vector bundle with the number of vector components in the previous models. Usually raising the dimensionality of vector representations is used to achieve better approximation and numerical stability of algorithms. Whether it is more appropriate to raise the dimensionality of the model (i. e., the rank of a vector bundle) or the number of harmonics used for approximation to achieve these goals in our case is an open question.

Another way of making the model more expressive is to incorporate other forms of dependent types, which are considered in type theoretical semantics, namely dependent sums and dot types. In the framework of Luo (2012) an adjective-noun pair is treated as an element of a dependent sum $\Sigma(x : A)B(x)$, where a noun is substituted for the x, e. g. a representation for *heavy book* is an object of type $\Sigma([\![\textit{book}]\!], [\![\textit{heavy}]\!])$. Objects of this type are pairs (a, b), where a is a noun and b is a variant of an adjective adapted to the noun similarly to the way the verbs are adapted to their objects. In the geometric interpretation such a pair is an element of a direct product of the base space and the fiber over a point in the base. Since the elements of A and the elements of $\Sigma(x : A)B(x)$ are not directly comparable, a mechanism of type coercion is required to make such a construct work.

6 Conclusion

In this paper we proposed a geometric interpretation for a fragment of lexical semantics based on dependent type theory. The fragment includes the dependent product, which is the type of functions with a range dependent on the argument, therefore the fragment also includes function types of traditional formal semantics as a special case. The types are interpreted as vector spaces, which makes it possible to treat vector models of distributional semantics as a computational realization of a fragment of type-theoretical semantics. By making extensions suggested by the geometric interpretation, we achieve a significant increase in expressive power of the model while retaining control over its computability. The meaning evaluation technique arising from the geometric interpretation is compositional and allows for an analysis of non-trivial phenomena of logical polysemy and co-predication.

References

Nicholas Asher and James Pustejovsky. 2006. A type composition logic for generative lexicon. *Journal of Cognitive Science*, 6:1–38.

Nicholas Asher. 2011. *Lexical meaning in context: A web of words*. Cambridge University Press.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology*, 9.

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

Robin Cooper. 2005. Records and record types in semantic theory. *Journal of Logic and Computation*, 15(2):99–112.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906. Association for Computational Linguistics.

Katrin Erk. 2009. Representing words as regions in vector space. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 57–65. Association for Computational Linguistics.

Peter Gärdenfors. 2004. *Conceptual spaces: The geometry of thought*. MIT press.

Patrick Hanks. 1996. Contextual dependency and lexical sets. *International Journal of Corpus Linguistics*, 1(1):75–98.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *CoNLL*, pages 114–123.

Ran Lahav. 1989. Against compositionality: the case of adjectives. *Philosophical studies*, 57(3):261–279.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Alessandro Lenci. 2008. Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics*, 20(1):1–31.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Percy Liang and Christopher Potts. 2015. Bringing machine learning and compositional semantics together. *Annu. Rev. Linguist.*, 1(1):355–376.

Glenys Luke and Alexander S. Mishchenko. 2013. *Vector bundles and their applications*, volume 447. Springer Science & Business Media.

Zhaohui Luo, Sergei Soloviev, and Tao Xue. 2013. Coercive subtyping: theory and implementation. *Information and Computation*, 223:18–42.

Zhaohui Luo. 2012. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513.

Bruno Mery, Christian Bassac, and Christian Retoré. 2007. A montagovian generative lexicon. In *12th conference on Formal Grammar (FG 2007)*. CSLI Publications.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Richard Montague. 1975. *Formal philosophy*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

James Pustejovsky. 1991. The generative lexicon. *Computational linguistics*, 17(4):409–441.

James Pustejovsky. 1995. *The generative lexicon.* Cambridge MA: MIT Press.

James Pustejovsky. 2013. Type theory and lexical decomposition. In *Advances in generative lexicon theory*, pages 9–38. Springer.

Aarne Ranta. 1994. *Type-theoretical grammar.*

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics*, 24(1):97–123.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1):159–216.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Kenneth N. Stevens. 1972. The quantal nature of speech: Evidence from articulatory-acoustic data. *Human communication: A unified view*, pages 51–66.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957. Association for Computational Linguistics.

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

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, and Yuji Matsumoto. 2013. Modeling and learning semantic co-compositionality through prototype projections and neural networks. In *EMNLP*, pages 130–140.

The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Conference of the North American Chapter of the Association of Computational Linguistics (HTL-NAACL)*, pages 1142–1151.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction*, volume 26.

Luke S. Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.