Static and Dynamic Vector Semantics for Lambda Calculus Models of Natural Language

Mehrnoosh Sadrzadeh and Reinhard Muskens

School of Electronic Engineering and Computer Science,
Queen Mary University of London.
mehrnoosh.sadrzadeh@qmul.ac.uk
Department of Philosophy, Tilburg University.
r.a.muskens@gmail.com

Abstract. Vector models of language are based on the contextual aspects of language, the distributions of words and how they co-occur in text. Truth conditional models focus on the logical aspects of language, compositional properties of words and how they compose to form sentences. In the truth conditional approach, the denotation of a sentence determines its truth conditions, which can be taken to be a truth value, a set of possible worlds, a context change potential, or similar. In the vector models, the degree of co-occurrence of words in context determines how similar the meanings of words are. In this paper, we put these two models together and develop a vector semantics for language based on the simply typed lambda calculus models of natural language. We provide two types of vector semantics: a static one that uses techniques familiar from the truth conditional tradition and a dynamic one based on a form of dynamic interpretation inspired by Heim's context change potentials. We show how the dynamic model can be applied to entailment between a corpus and a sentence and provide examples.

Keywords. Vector semantics · Simply typed lambda calculus · Context update · Context change potential · Compositionality.

1 Introduction

Vector semantic models, otherwise known as distributional models, are based on the contextual aspects of language, the company each word keeps, and patterns of use in corpora of documents. Truth conditional models focus on the logical and denotational aspects of language, how words can be represented by functions over sets and application and composition of these functions. Vector semantics and truth conditional models are based on different philosophies; one takes the stance that language is contextual, the other asserts that it is logical. In recent years there has been much effort to bring these two together. We have models that are based on a certain type of grammatical representation, e.g. the pregroup model of [4], the Lambek Calculus model of [5], and the combinatorial categorial models of [15,17]; we also have more concrete models that draw inspirations from type theory but whose major development is developing concrete ways of constructing linear and multi linear algebraic counterparts for the syntactic types, e.g. matrices and tensors of [8,2] and relational clusters of [16].

What these approaches, [4,15,17] more than [2,16], miss is acknowledging the inherent gap between the contextual and truth conditional semantics; they closely follow the truth theoretic conditions to assign vector representations to phrases and sentences. Indeed it is possible to develop a stand-alone compositional vector semantics along these lines, but this will result in a static semantics. From the perspective of the underlying theory it will also be quite natural to have such a vector semantics work in tandem with a dynamic theory and let the two modules model different aspects of meaning. Distributional semantics is particularly apt at modelling associative aspects of meaning, while truth-conditional and dynamic forms of semantics are good at modelling the relation of language to reality and at modelling entailment. It is quite conceivable that a theory that combines the two as separate modules will be simpler than one that tries to make one of the approaches do things it was never intended for.

This is what we will do in this paper. We first sketch how an approach to semantics that in many of its aspects derives from the one pioneered by [19] can be used to assign vector meanings to linguistic phrases. The theory will be based on the simply typed lambda calculus and as a result will be neutral with respect to the linguist's choice of syntax, in the sense that it can be combined with any existing syntax-semantics interface that assumes that the semantics is based on lambdas (e.g. linguistic trees + 'type-driven translation', f-structures + 'glue' in Lexical-Functional Grammar, derivations in Combinatory Categorial Grammar + the use of combinators, proofs + 'semantic recipes' in Lambek Categorial Grammar, etc.).

Our reasoning for the use of lambda calculus is that it directly relates our semantics to higher order logic and makes standard ways of treating long distance dependencies and coordination accessible to vector-based semantics. This approach results in the same semantics as the static approaches listed above. The reason for providing it is to show that a lambdas calculus model of language can directly be provided with a straightforward vector semantics. As will be seen, abstract lambda terms, which can be used as translations of linguistic expressions have a lot in common with the Logical Forms of these expressions and the lambda binders in them give easy ways for treating long distance dependencies. The use of lambda terms also makes standard ways of dealing with coordination accessible to distributional semantics. We provide extensive discussion of this and will give examples where the direct use of lambdas gives an edge over the above listed static approaches.

The above semantics does not have an explicit notion of context however. The second contribution of this paper paper is that, based on the same lambda calculus model of natural language, we develop a dynamic vector interpretation for this type theory where denotations of sentences are "context change potentials", as introduced by Heim [11]. We show how to assign such a vector interpretation to words and how these interpretations compose in a way such that the vectors of the sentences containing them change the context in a dynamic style similar to the one instructed by Heim. A context can be interpreted in different ways, we work with two different notions of context in distributional semantics: co-occurrence matrices and entity relation graphs, encoded here in the form of cubes. Both of these are built from corpora of documents and record the co-occurrence between the words: in a simple neighbourhood window for the case of co-occurrence matrices and in a window structured by grammatical dependencies, in

the case of an entity relation cube. We believe our model is flexible enough with other distributional notions of contexts, such as networks of grammatical dependancies. We show how our approach relates to Heim's original notion of 'files' as contexts. Other dynamic approaches, such the update semantics of [30] and the continuation-based semantics of [9], can also be used; we aim to do this in the future.

We are after a compositional vector semantics, but this paper is theoretical. So what we will not do is settle upon—from the armchair so to speak—certain concrete representations of contexts and updates and a set of concrete vector composition operations for combining phrases or concrete matrices or cubes that embody them. We thus will leave an exhaustive empirical evaluation of our model to future work, but show, by means of examples, how the notion of "admittance of sentences by contexts" from the context update logic of Heim and Karttunen can be applied to develop a relationship between matrix and cube contexts and sentences and how this notion can be extended from a usual boolean relation to one which has degrees, based on the notion of degrees of similarity between the words. This notion resembles that of a "contextual entailment" between corpora and sentences, we review the current entailment datasets that are mainstream in distributional semantics and discuss how they can or cannot be applicable to test this notion. but leave an experimental valuation to future work.

The lambda calculus approach we use is based on the Lambda Grammars of [21,22], which were independently introduced as Abstract Categorial Grammars (ACGs) in [10]. The theory developed here, however, can be based on any syntax-semantics interface that works with a lambda calculus based semantics. Our approach is agnostic as to the choice of a syntactic theory. Lambda Grammars/ACGs are just a framework for thinking about type and term homomorphisms and we are using them entirely in semantics here.

This paper is the journal version of our previous extended abstract [25] and short paper [24].

2 Lambda Grammars

The Lambda Grammars of [21,22] were independently introduced as Abstract Categorial Grammars (ACGs) in [10]. An ACG generates two languages, an *abstract* language and an *object* language. The abstract language will simply consist of all linear lambda terms (each lambda binder binds exactly one variable occurrence) over a given vocabulary typed with *abstract types*. The object language has its own vocabulary and its own types. We give some basic definitions here, assuming familiarity with the simply typed λ-calculus.

If $\mathcal{B}$ is some set of basic types, we write $\mathit{TYP}(\mathcal{B})$ for the smallest set containing $\mathcal{B}$ such that $(\alpha\beta) \in \mathit{TYP}(\mathcal{B})$ whenever $\alpha, \beta \in \mathit{TYP}(\mathcal{B})$. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be sets of basic types. A function η from $\mathit{TYP}(\mathcal{B}_1)$ to $\mathit{TYP}(\mathcal{B}_2)$ is said to be a *type homomorphism* if $\eta(\alpha\beta) = (\eta(\alpha)\eta(\beta))$, for all $\alpha, \beta \in \mathit{TYP}(\mathcal{B}_1)$. It is clear that a type homomorphism η with domain $\mathit{TYP}(\mathcal{B})$ is completely determined by the values of η for types $\alpha \in \mathcal{B}$.

Let us look at an example of a type homomorphism that can be used to provide a language fragment with a classical Montague-like meaning. Let $\mathcal{B}_1 = \{D, N, S\}$ (D stands for determiner phrases, N for nominal phrases, S for sentences), let $\mathcal{B}_2 = \{e, s, t\}$ (e is for entities, s for worlds, and t for truth-values), and let h_0 be defined by:

constant c	type τ	$H_0(c)$	$h_0(\tau)$
`woman`	N	*woman*	est
`man`	N	*man*	est
`tall`	NN	*tall*	$(est)est$
`smokes`	DS	*smoke*	est
`loves`	DDS	*love*	$eest$
`knows`	SDS	$\lambda p\lambda x\lambda w.\forall w'(Kxww' \rightarrow pw')$	$(st)est$
`every`	$N(DS)S$	$\lambda P'\lambda P\lambda w.\forall x(P'xw \rightarrow Pxw)$	$(est)(est)st$
`a`	$N(DS)S$	$\lambda P'\lambda P\lambda w.\exists x(P'xw \wedge Pxw)$	$(est)(est)st$

Table 1. An Abstract Categorial Grammar / Lambda Grammar connecting abstract terms with Montague-like meanings. Here p is a variable of type st, while x is of type e, w and w' are of type s, and P and P' are of type est. The constant K of type ess denotes the epistemic accessibility relation.

$h_0(D) = e$, $h_0(N) = est$,[1] and $h_0(S) = st$. Then the types in the second column of Table 1 have images under h_0 as given in the fourth column.

We now define the notion of a *term homomorphism*. If C is some set of typed constants, we write $\Lambda(C)$ for the set of all lambda terms with constants only from C. The set of *linear* lambda terms over C is denoted by $\Lambda_0(C)$. Let C_1 be a set of constants typed by types from $\mathit{TYP}(\mathcal{B}_1)$ and let C_2 be a set of constants typed by types from $\mathit{TYP}(\mathcal{B}_2)$. A function $\vartheta : \Lambda(C_1) \rightarrow \Lambda(C_2)$ is a *term homomorphism based on* η if $\eta : \mathit{TYP}(\mathcal{B}_1) \rightarrow \mathit{TYP}(\mathcal{B}_2)$ is a type homomorphism and, whenever $M \in \Lambda(C_1)$:

- $\vartheta(M)$ is a term of type $\eta(\tau)$, if M is a constant of type τ;
- $\vartheta(M)$ is the n-th variable of type $\eta(\tau)$, if M is the n-th variable of type τ;
- $\vartheta(M) = (\vartheta(A)\vartheta(B))$, if $M \equiv (AB)$;
- $\vartheta(M) = \lambda y.\vartheta(A)$, where $y = \vartheta(x)$, if $M \equiv (\lambda x.A)$.

Note that this implies that $\vartheta(M)$ is a term of type $\eta(\tau)$, if M is of type τ.

Clearly, a term homomorphism ϑ with domain $\Lambda(C)$ is completely determined by the values $\vartheta(c)$ for $c \in C$. This continues to hold if we restrict the domain to the set of linear lambda terms $\Lambda_0(C)$.

In order to show how this can be used, let us continue the example we just looked at. Consider the (abstract) constants in the first column of Table 1, typed by the (abstract) types in the second column. We can now define a term homomorphism H_0 by sending the constants in the first column to their images in the third column, making sure that these have types as in the fourth. Since H_0 is assumed to be a type homomorphism, *all* lambda terms over the constants in the first column will now automatically have images

[1] Association in types is to the right and outer parentheses are omitted; so est is short for $(e(st))$, arguably a good type for *predicates*.

under H_0. For example, it can be easily checked that H_0 sends the abstract term[2]

$$((\texttt{a woman})\lambda\xi((\texttt{every man})(\texttt{loves } \xi)))$$

(in which ξ is of type D), to a term $\beta\eta$-equivalent with

$$\lambda w \exists y(woman\, yw \wedge \forall x(man\, xw \rightarrow love\, yxw)) \; .$$

This term denotes the set of worlds in which some specific woman is loved by all men.

While this example sends abstract terms to translations that are close to the ones of [19] and while such translations obviously will not do as a *vector* semantics, we will show in the next sections that it is possible to alter the object language while retaining the general translation mechanism. For more information about the procedure of obtaining an object language from an abstract language, see the papers mentioned or the explanation in [23].

3 A Static Vector Semantics

3.1 Vector Interpretations for the Object Language

In order to provide an interpretation of our object language, the type theory used must be able to talk about vectors over some field, for which we choose the reals. We need a basic object type R such that, in all interpretations under consideration, the domain D_R of type R is equal to or 'close enough' to the set of reals $\mathbb{R}$ and such that constants such as the following have their usual interpretation (the $\rightarrow$ in types is dropped and association is to the right; constants such as $+$, $\cdot$, and $<$ will be written between their arguments).

$$\begin{aligned} 0 &: R \\ 1 &: R \\ + &: RRR \\ \cdot &: RRR \\ < &: RRR \end{aligned}$$

This can be done by imposing one of the sets of second-order axioms in [29]. Given these axioms we have that $D_R = \mathbb{R}$ in full models, while we get non-standard models under the Henkin interpretation.

Vectors can now be introduced as objects of type IR, where I is interpreted as some finite index set. Think of I as a set of words; if a word is associated with a vector $v : IR$, v assigns a real number to each word, which gives information about the company the word keeps. Since IR will be used often, we will abbreviate it as V. Similarly, IIR, abbreviated as M, can be associated with the type of *matrices* and $IIIR$, abbreviated as C, with the type of *cubes*, and so on, see Table 2.

[2] We use the standard notation of lambda terms. The application of M to N is written as (MN) (not as $M(N)$) and lambda abstractions are of the form $(\lambda X.A)$. The usual redundancy rules for parentheses apply, but will often not be used in abstract terms, in order to emphasise their closeness to linguistic expressions. In some cases, where it seems to improve clarity, we will flout the rules and write things like $M(N_1, \ldots, N_n)$ for $(MN_1 \ldots N_n)$ or $A \wedge B$ for $\wedge AB$.

Type	Abbreviation	
IR	V	Vector
IIR	M	Matrix
$IIIR$	C	Cube
$IIIIR$	H	Hypercube
$\cdots$		

Table 2. Vector types and their abbreviations

In this paper we will work with a single index type, but in general one can also consider cases with several index types, so that phrases of distinct categories can live in their own space.

We need a toolkit of functions combining vectors, matrices, cubes, etc. Here are some definitions, in which r is of type R; i, j, and k are of type I; v and u are of type V; and m and c are of types M and C respectively. Indices are written as subscripts—v_i is syntactic sugar for vi.

$$
\begin{aligned}
* &:= \lambda rvi.r \cdot v_i : RVV \\
\boxplus &:= \lambda vui.v_i + u_i : VVV \\
\odot &:= \lambda vui.v_i \cdot u_i : VVV \\
\times_1 &:= \lambda mvi. \sum_j m_{ij} \cdot v_j : MVV \\
\times_2 &:= \lambda cvij. \sum_k m_{ijk} \cdot v_k : CVM \\
\langle \cdot \mid \cdot \rangle &:= \lambda uv. \sum_i u_i + v_i : VVR
\end{aligned}
$$

The reader will recognise $*$ as scalar product, $\boxplus$ as pointwise addition, $\odot$ as pointwise multiplication, $\times_1$ and $\times_2$ as matrix-vector and cube-vector multiplication, and $\langle \cdot \mid \cdot \rangle$ as the dot product. One can also consider further operations, such as a *rotation* operation $\rho : VVV$, given below

$$
\rho(\cdot,\cdot) := \lambda vu. \begin{pmatrix} \langle u \mid v \rangle & \pm\sqrt{1 - \langle u \mid v \rangle^2} \\ \pm\sqrt{1 - \langle u \mid v \rangle^2} & \langle u \mid v \rangle \end{pmatrix} \times \frac{u+v}{2}
$$

This operation takes two vectors and produces the result of rotating their average towards the first vector. This can be thought of as, for example, providing a good candidate for modelling head-argument combinations. We will not present a specific use for this operation in this paper and are just introducing it as an example of a new operation that can be, but has not been, used by the existing vector models, in order to illustrate the range of vector operation that can be modelled in our setting.

3.2 Abstract Types and Type and Term Homomorphisms

Let us assume again that our basic abstract types are D for determiner phrases, S for sentences, and N for nominal phrases. But this time our type and term homomorphisms will be chosen in a way different from how it was done in section 2. A very simple type homomorphism h can be defined by letting

$$h(D) = h(S) = h(N) = V$$

So h assigns vectors to determiners, nominal phrases and sentences. There are other possibilities for the range of h and we will sketch a more elaborate assignment in which a running context is used in the next section. The above simple h is chosen for the expository purposes of this section.

In Table 3, we again provide abstract constants in the first column and their abstract types in the second column; h assigns to these the object types in the fourth column. For instance, the constant `woman` has the abstract type N, and a term homomorphic image woman, which is assigned the type V by h. We say that the translation of `woman` is of type V. Similarly, the translations of `tall` and `smoke` are of type VV, `love` and `know` are of type VVV, and those of `every`, and `a` are of type VV. The term homomorphism H is defined by letting its value for any abstract constant in the first column be the corresponding object term in the third column. Using this table, we automatically obtain homomorphic images of any lambda term over the constants. But now our previous example term[3]

$$((\texttt{a woman})\lambda\xi((\texttt{every man})(\texttt{loves } \xi)))$$

is sent to a term that is $\beta\eta$ equivalent with

$$(\mathsf{love} \times_2 (\mathsf{a} \times_1 \mathsf{woman})) \times_1 (\mathsf{every} \times_1 \mathsf{man}) .$$

Nominal phrases like woman are represented by vectors in Table 3, adjectives and intransitive verbs like tall and smoke by matrices, and transitive verbs (love) by cubes, as are constants like know. Generalised quantifiers are functions that take vectors to vectors. The composition operations used ($\times_1$ and $\times_2$) are cube-vector and matrix-vector instances of tensor contraction. The jury is still very much out on what are the best operations for composing vectors. [18] consider pointwise addition and multiplication of vectors, matrix multiplication is used in [1]. Such operations are available to our theory. The table for these will have a different $H(c)$ column and will be the same in all the other columns. The $H(c)$ columns for these models are given in Table 4.

In this paper, we will not choose between these operations. Instead of this we will explore the question how to combine such functions once an initial set of them has been established (and validated empirically). Functions in the initial set will typically combine vector meanings of adjacent phrases. Our aim, like the one in [2] (who also give an excellent introduction to and survey of the work that has been done in compositional vector semantics), has been to give a general theory that also includes dependencies between phrases that are not adjacent, such as in topicalisation and relative clause formation.

[3] The entry for `man` is no longer present in Table 3. But `man` can be treated in full analogy to `woman`. In further examples we will also use constants whose entries can be easily guessed.

c	τ	$H(c)$	$h(\tau)$
`woman`	N	woman	V
`tall`	NN	$\lambda v.(\mathsf{tall} \times_1 v)$	VV
`smokes`	DS	$\lambda v.(\mathsf{smoke} \times_1 v)$	VV
`loves`	DDS	$\lambda uv.(\mathsf{love} \times_2 u) \times_1 v$	VVV
`knows`	SDS	$\lambda uv.(\mathsf{know} \times_2 u) \times_1 v$	VVV
`every`	$N(DS)S$	$\lambda vZ.Z(\mathsf{every} \times_1 v)$	$V(VV)V$
`a`	$N(DS)S$	$\lambda vZ.Z(\mathsf{a} \times_1 v)$	$V(VV)V$

Table 3. Some abstract constants c typed with abstract types τ and their term homomorphic images $H(c)$ typed by $h(\tau)$. Here, Z is a variable of type VV, and v and u are of type V.

4 Dynamic Vector Semantics with Context Change Potentials

4.1 Heim's Files and Distributional Contexts

Heim describes her contexts as files that have some kind of information written on (or in) them. Context changes are operations that update these files, e.g. by adding or deleting information from the files. Formally, a context is taken to be a set of sequence-world pairs, in which the sequences come from some domain $\mathcal{D}_I$ of individuals, as follows:

$$ctx \subseteq \{(g, w) \mid g\colon \mathbb{N} \to \mathcal{D}_I, w \text{ a possible world}\}$$

We follow Heim [11] here in letting the sequences in her sequence-world-pairs be infinite, although they are best thought of as finite.

Sentence meanings are *context change potentials* (CCPs) in Heim's work, functions from contexts to contexts. A sentence S comes provided with a sequence of instructions that, given any context ctx, updates its information so that a new context denoted as

$$ctx + S$$

results. The sequence of instructions that brings about this update is derived compositionally from the constituents of S.

In distributional semantics, contexts are words somehow related to each other via their patterns of use, e.g. by co-occurring in a neighbourhood word window of a fixed size or via a dependency relation. In practice, one builds a context matrix M over $\mathbb{R}^2$, with rows and columns labeled by words from a vocabulary Σ and with entries taking values from $\mathbb{R}$, for a full description see ([28]). M can be seen as the set of its vectors:

$$\{\overrightarrow{v} \mid \overrightarrow{v}\colon \Sigma \to \mathbb{R}\}$$

where each $\overrightarrow{v}$ is a row or column in M.

Addition	Multiplication	Matrix Multiplication
$H(c)$	H(c)	H(c)
woman	woman	woman
$\lambda v.(\mathsf{tall} \boxplus v)$	$\lambda v.(\mathsf{tall} \odot v)$	$\lambda v.(\mathsf{tall} \times_1 v)$
$\lambda v.(\mathsf{smoke} \boxplus v)$	$\lambda v.(\mathsf{smoke} \odot v)$	$\lambda v.(\mathsf{smoke} \times_1 v)$
$\lambda uv.(\mathsf{love} \boxplus u) \boxplus v$	$\lambda uv.(\mathsf{love} \odot u) \odot v$	$\lambda uv.(\mathsf{love} \times_1 u) \times_1 v$
$\lambda uv.(\mathsf{know} \boxplus u) \boxplus v$	$\lambda uv.(\mathsf{know} \odot u) \odot v$	$\lambda uv.(\mathsf{know} \times_1 u) \times_1 v$
$\lambda vZ.Z(\mathsf{every} \boxplus v)$	$\lambda vZ.Z(\mathsf{every} \odot v)$	$\lambda vZ.Z(\mathsf{every} \times_1 v)$
$\lambda vZ.Z(\mathsf{a} \boxplus v)$	$\lambda vZ.Z(\mathsf{a} \odot v)$	$\lambda vZ.Z(\mathsf{a} \times_1 v)$

Table 4. Term homomorphic images $H(a)$ for pointwise addition and multiplication, and matrix multiplication as composition operations. In the case of addition and multiplication, Z is a variable of type V, for matrix multiplication it is of type VV; whereas v and c are of type V for all three operations.

If we take Heim's domain of individuals $\mathcal{D}_I$ be the vocabulary of a distributional model of meaning, that is $\mathcal{D}_I := \Sigma$, then a context matrix can be seen as a so-called *quantized* version of a Heim context:

$$\{(\overrightarrow{g}, w) \mid \overrightarrow{g} \colon \Sigma \to \mathbb{R}, w \text{ a possible world}\}$$

Thus a distributional context matrix is obtainable by endowing Heim's contexts with $\mathbb{R}$. In other words, we are assuming that not only a file has a set of individuals, but also that these individuals take some kind of values, e.g. from reals.

The role of possible worlds in a distributional semantics is arguable, as vectors retrieved from a corpus are not naturally truth conditional. Keeping the possible worlds in the picture provides a machinery to assign a proposition to a distributional vector by other means and can become very useful. We leave working with possible worlds to future work and in this paper only work with the sets of vectors as our contexts, that is:

$$ctx \subseteq \{\overrightarrow{g} \mid \overrightarrow{g} \colon \Sigma \to \mathbb{R}, g \in M\} \tag{1}$$

Distributional versions of Heim's CCP's can be defined based on the intuitions and definitions of Heim. In what follows we pan out how these instructions let contexts thread through vectorial semantics in a compositional manner.

4.2 Dynamic Type and Term Homomorphisms and their Interpretations

On the set of basic abstract types D, S, N a *dynamic* type homomorphism ρ that takes into account the contexts of words is defined as follows:

$$\rho(N) = (VU)U, \quad \rho(D) = V, \quad \rho(S) = U$$

Here, sentences are treated as *context change potentials*. They update contexts and we assign the type U (for 'update') to them. A context can be a matrix or a cube, so it can have type I^2R or I^3R. A sentence can then have type $(I^2R)(I^2R)$ or $(I^3R)(I^3R)$. We have previously abbreviated IR to V, I^2R to M, and I^3R to C. The sentence type then becomes MM or CC, used as an abbreviation for U; the former in the context matrix setting and the latter in the context cube setting. The concrete semantics obtained by instantiating U to matrix and cube contexts are discussed in more details in sections 5 and 6, respectively.

The update functions are presented in Table 5. Simple words such as names, nouns, adjectives, and verbs are first assigned vectors, denoted by constants such as anna, woman, tall and smoke (all of type V). These are then used by the typed lambda calculus given via $H(a)$, in the third column, to build certain functions, which will act as the meanings of those words in context. The object types assigned by ρ are as follows:

Type of nouns	: $(VU)U$
Type of adjectives	: $((VU)U)(VU)U$
Type of intransitive verbs	: VU
Type of transitive verbs	: VVU

The function Z updates the context of proper names and nouns based on their vectors e.g. anna and woman. These are essentially treated as vectors with type V, but, since they must be made capable of dynamic behaviour, they are 'lifted' to the higher type $(VU)U$.

The function F of an adjective, takes a vector for the adjective, e.g. tall, a vector for its argument, e.g. v and a vector for its context, e.g. c, then updates the context, e.g. as in $F(\mathsf{tall}, v, c)$. The output of this function is then lifted to the the higher type, i.e. $((VU)U)((VU)U)$ via the functions Z and Q, respectively.

Functions G and I update contexts of verbs; they take a vector for the verb as well as a vector for each of its arguments, plus an input context, and then return a context as their output. So, the function G of an intransitive verb takes a vector for itself, e.g. smoke a vector for its subject, e.g. v, plus a context, e.g. c, and returns a modified context, e.g. via $\lambda vc.G(\mathsf{smoke}, v, c)$. The function I of a transitive verb takes a vector for itself, a vector for its subject, a vector for its object and a context, and returns a context.

The meanings of function words, such as conjunctions, relative pronouns, and quantifiers, will not (necessarily) be identified with vectors. The type of the quantifier *every* is $((VU)U)(VU)U$, where its noun argument has the required 'quantifier' type $(VU)U$. The lambda calculus entry for 'every', $\lambda Q.Q$, is the identity function; it takes a Q and then spits it out again. The alternative would be to have an entry along the lines of that of 'tall', but this would not make a lot of sense. It is the content words that seem to be important in a distributional setting, not the function words.

The word *and* is treated as a generalised form of function composition. Its entry is schematic, as *and* does not only conjoin sentences, but also other phrases of any category. So, the type of the abstract constant connected with the word is $(\overline{\alpha}S)(\overline{\alpha}S)(\overline{\alpha}S)$, in

a	τ	$H(a)$	$\rho(\tau)$
`Anna`	$(DS)S$	$\lambda Z.Z(\mathsf{anna})$	$(VU)U$
`woman`	N	$\lambda Z.Z(\mathsf{woman})$	$(VU)U$
`tall`	NN	$\lambda QZ.Q(\lambda vc.ZvF(\mathsf{tall}, v, c))$	$((VU)U)(VU)U$
`smokes`	DS	$\lambda vc.G(\mathsf{smoke}, v, c)$	VU
`loves`	DDS	$\lambda uvc.I(\mathsf{love}, u, v, c)$	VVU
`knows`	SDS	$\lambda pvc.pJ(\mathsf{know}, v, c)$	UVU
`every`	$N(DS)S$	$\lambda Q.Q$	$((VU)U)(VU)U$
`who`	$(DS)NN$	$\lambda Z'QZ.Q(\lambda vc.Zv(QZ'c))$	$(VU)((VU)U)(VU)U$
`and`	$(\overline{\alpha}S)(\overline{\alpha}S)(\overline{\alpha}S)$	$\lambda R'\lambda R\lambda\overline{X}\lambda c.R'\overline{X}(R\overline{X}c)$	$(\rho(\overline{\alpha})U)(\rho(\overline{\alpha})U)(\rho(\overline{\alpha})U)$

Table 5. Some abstract constants a typed with abstract types τ and their term homomorphic images $H(a)$ typed by $\rho(\tau)$ (where ρ is a type homomorphism, i.e. $\rho(AB) = \rho(A)\rho(B)$). Here Z is a variable of type VU, Q is of type $(VU)U$, v of type V, c of type M, and p and q are of type U. The functions F, G, I, and J are explained in the main text. In the schematic entry for `and`, we write $\rho(\overline{\alpha})$ for $\rho(\alpha_1)\cdots\rho(\alpha_n)$, if $\overline{\alpha} = \alpha_1\cdots\alpha_n$.

which $\overline{\alpha}$ can be any sequence of abstract types. Ignoring this generalisation for the moment, we obtain SSS as the abstract type for sentence conjunction, with a corresponding object type UUU, and meaning $\lambda pqc.p(qc)$, which is just function composition. This is defined in a way such that the context updated by *and*'s left argument will be further updated by its right argument. So 'Sally smokes and John eats bananas' will, given an initial context c, first update c to $G(\mathsf{Sally}, \mathsf{smoke}, c)$, which is a context, and then update further with 'John eats bananas' to $I(\mathsf{eat}, \mathsf{John}, \mathsf{bananas}, G(\mathsf{smoke}, \mathsf{Sally}, c))$. This treatment of *and* is easily extended to coordination in all categories. For example, the reader may check that `and admires loves` (which corresponds to *loves and admires*) has $\lambda uvc.I(\mathsf{admire}, u, v, I(\mathsf{love}, u, v, c))$ as its homomorphic image.

The update instructions fall through the semantics of phrases and sentences compositionally. The sentence *every tall woman smokes*, for example, will be associated with the following lambda expression:

```
(((every (tall woman)) smokes))
```

This in its turn has a term homomorphic image that is β-equivalent with the following:

$$\lambda c.G\,(\mathsf{smoke}, \mathsf{woman}, F(\mathsf{tall}, \mathsf{woman}, c))$$

which describes a distributional context update for it. This term describes a first update of the context c according to the rule for the constant `tall`, and then a second update according to the rule for the constant `smokes`. As a result of these, the value entries

at the crossings of $\langle$tall, woman$\rangle$ and $\langle$woman, smokes$\rangle$ get increased. Much longer chains of context updates can be 'threaded' in this way.

In the following, we give some examples. In each case the a. sentence is followed by an abstract term in b. which captures its syntactic structure. The update potential that follows in c. is the homomorphic image of this abstract term.

(1) a. Sue loves and admires a stockbroker
b. $\texttt{(a stockbroker)}\lambda\xi.\texttt{Sue(and admires loves }\xi\texttt{)}$
c. $\lambda c.I(\mathsf{admire}, \mathsf{stockbroker}, \mathsf{sue}, I(\mathsf{love}, \mathsf{stockbroker}, \mathsf{sue}, c))$

(2) a. Bill admires but Anna despises every cop
b. $\texttt{(every cop)}\lambda\xi.\texttt{and(Anna(despise }\xi\texttt{))(Bill(admire }\xi\texttt{))}$
c. $\lambda c.I(\mathsf{despise}, \mathsf{cop}, \mathsf{anna}, I(\mathsf{admire}, \mathsf{cop}, \mathsf{bill}, c))$

(3) a. The witch who Bill claims Anna saw disappeared
b. $\texttt{the(who(}\lambda\xi.\texttt{Bill(claims(Anna(saw }\xi\texttt{)))witch)disappears}$
c. $\lambda c.G(\mathsf{disappear}, \mathsf{witch}, I(\mathsf{see}, \mathsf{witch}, \mathsf{anna}, J(\mathsf{claim}, \mathsf{bill}, c)))$

5 Co-occurence Matrix Context and its Update

In this section we assume that our contexts are the co-occurrence matrices of distributional semantics [28]. Given a corpus of text, a co-occurrence matrix has at each of its entries the degree of co-occurrence between a word and its neighbouring words. The neighbourhood is usually a window of k words to both sides of the word. The update type U associated to sentences, will thus take the form $(I^2R)(I^2R)$, abbreviated to MM. That is, a sentence will take a co-occurrence matrix as input, update it with new entries, and return the updated matrix as output.

Since we are working with co-occurrence matrices, the updates simply increase the degrees of co-occurrences between the labelling words of the rows and columns of the matrix. In this paper, for the purpose of keeping it simple, we work with a co-occurrence matrix with raw co-occurrence numbers as entries. In this case, the update functions just add 1 to each entry at each single update step. This may be extended to (or replaced with) logarithmic probabilistic entries such as Pointwise Mutual Information (PMI) or its positive or smoothed version PPMI, PPMI_α, in which case the update functions have to recalculate these weighting schemes at each step. For instance, see the example presented in Table 6. The cells whose entries are increased are chosen according to the grammatical roles of the labelling words. These are implemented in the functions F, G, I, J, which apply the updates to each word of the sentence. The updates are compositional, i.e. they can be applied compositionally to the words within a sentence. This is evident as the updates induced by words of a sentence are designed based on the grammatical roles of them and this acts a glue.

More formally, the object terms corresponding to a word a update a context matrix c with the information in a and the information in the vectors of arguments $u, v, \cdots$ of a. The result is a new context matrix c', with different value entries, depicted below:

$$\begin{pmatrix} m_{11} & \cdots & m_{1k} \\ m_{21} & \cdots & m_{2k} \\ \vdots & & \\ m_{n1} & \cdots & m_{nk} \end{pmatrix} + \langle a, u, v, \cdots \rangle = \begin{pmatrix} m'_{11} & \cdots & m'_{1k} \\ m'_{21} & \cdots & m'_{2k} \\ \vdots & & \\ m'_{n1} & \cdots & m'_{nk} \end{pmatrix}$$

where we have

$$m'_{ij} := m_{ij} + 1$$

The m_{ij} and m'_{ij} entries are described as follows:

- The function denoted by $\lambda vc.G(\mathsf{smoke}, v, c)$ increases the value entry of m_{ij} of c by 1, for i and j indices of smoke and its subject v.
- The function denoted by $\lambda uv.\lambda c.I(\mathsf{love}, u, v, c)$ increases the value entries of m_{ij}, m_{jk}, and m_{ik} of c by 1, for i, j, k indices of loves, its subject u and its object v.
- The function denoted by $\lambda vc.F(\mathsf{tall}, v, c)$ increases the value entry of m_{ij} of c by 1, for i and j indices of tall and its modified noun v. The entry for *tall* in Table 1 uses this function, but allows for further update of context.
- The function denoted by $\lambda vc.J(\mathsf{know}, v, c)$ increases the value entry of m_{ij} of c by 1, for i and j indices of know and its subject v. The updated matrix is made the input for further update (by the context change potential of the sentence that is known) in Table 1.

As an example, consider the co-occurrence matrices depicted in Figure 1. The left hand side matrix is a snapshot of a matrix just before a series of updates are applied to it. In this matrix, for individual nouns such as `Anna` and noun phrases such as woman we assume Z is the identity. So their entries are the same as the entries of their distributional co-occurrence vectors. The rationale of this example for the entries of these words is as follows: `Anna` is a woman and so it has not much appeared in a corpus in the neighbourhood of the context `men`; as a result, it has a low value of 100 at that entry; she loves cats (and has some herself), so her entry at the context `cat` is 700; she loves other things such as smoking, and so she has a substantial entry at the context `loves`; and so on. The entries of the other words, i.e. `tall, smokes, loves, knows`, are also initialised to their distributional co-occurrence matrix vectors. When an entry c_{ij} corresponds to the same two words, e.g. when i and j are both `love` as in the left hand side matrix of Figure 1, we put a ϵ to indicate a predefined fixed value.

	1 man	2 cat	3 loves	4 fears	5 sleeps
1 Anna	100	700	800	500	400
2 woman	500	650	750	750	600
3 tall	300	50	500	400	400
4 smokes	400	50	600	600	200
5 loves	350	250	ϵ	600	500
6 knows	300	50	200	250	270

$\overset{\text{update by}}{\Longrightarrow}$ G, I, F, J

	1 man	2 cat	3 loves	4 fears	5 sleeps
1 Anna	100	700	800	500	400
2 woman	500	650	750	750	600
3 tall	650	50	500	400	400
4 smokes	700	50	600	600	200
5 loves	550	750	ϵ	600	500
6 knows	600	250	450	510	700

Fig. 1. An example of updates by functions F, G, I, J on a co-occurrence matrix

The intransitive verb `smokes` updates the left hand side matrix of Figure 1 via the function G at the entries c_{4j}. Here, in principle, j can be 1 and 2, as both `man` and `cat`, in their singular or plural forms, could have occurred as subjects of `smokes` in the corpus. Assuming that cats do not smoke and that a reasonable number of men do, a series of, for instance, 300 occurrences of `smokes` with the subject `man`, updates this entry and raises its value from 400 to 700. Similarly, adjective `tall` updates the entries of the c_{3j} cells of the matrix via the function F, where j can in principle be 1 and 2, but since cats are not usually tall, it only updates c_{31}. Again, a series of, for example, 350 occurrences of the adjective `tall` as the modifier of `man` raises this number from 300 to, say 650. The case for `loves` and function I is similar. For `knows`, men know cats love mice, and love to play and be stroked, etc; they know that cats fear water and objects such as vacuum cleaners, and that they sleep a lot. As a result, the values of all of the entires of row 6, that is $c_{61}, c_{62}, c_{63}, c_{64}$ and c_{65} will be updates by function J, for instance, to the numbers in the right hand side table.

6 Entity Relation Cube Context and its Update

A corpus of text can be seen as a sequence of lexical items occurring in the vicinities of each other and turned into a co-occurrence matrix. It can also be seen as a sequence of entities related to each other via predicate-argument structures and turned into an entity relation graph. These can be modelled in our setting by taking the contexts to be cubes, thus setting S to have the update type $U = (I^3R)(I^3R)$, abbreviated to CC. The entity relation graph approach needs a more costly preprocessing of the corpus, but it is useful for a systematic treatment of logical words such as negation and quantification, as well as coordination.

An entity relation graph can be derived from different resources: a semantic network of concepts, a knowledge base such as WordNet or FrameNet, or an olog or a Gellish network. We work with entities and relations extracted from text. Discovering such graphs from corpora of text in an automatic way has been subject of much recent research, for example see [31,27] for a direct approach and [13,26] via semantic parsing. The elements of an entity relation graph are argument-relation-argument triples, sometimes referred to by *relation paths* [31]. Similar to [16] we base ourselves in a binary

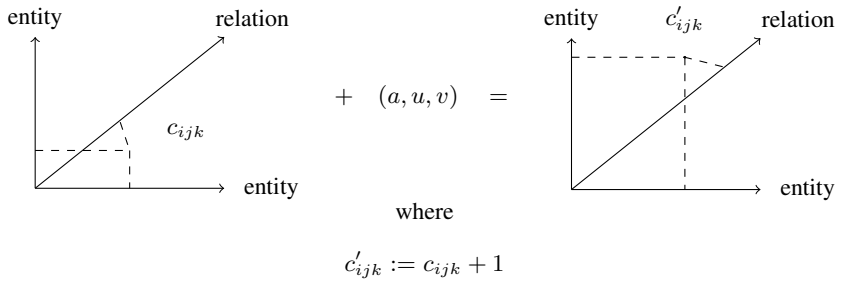

Fig. 2. Updates of entries in an entity relation cube

version of the world, where our relations are all binary; we turn non-binary relations into binary ones using the is-a predicate.

Similar to the matrix case, the object terms corresponding to a constant a update a context cube c with the information in a and the information in the vectors of arguments of a. The result is a new context cube c', with different value entries, greater or less than the originals, as depicted in Figure 2.

The c_{ijk} and c'_{ijk} entries are similar to the matrix case, for example

- The function denoted by $\lambda vc.G(\mathsf{smoke}, v, c)$ increases the value entry c_{ijk} of c, for i, j, k indices of is-a and smoker and v the subject of smoke.
- The function denoted by $\lambda vc.F(\mathsf{tall}, v, c)$ increases the value entry c_{ijk} of c, for i, j, k indices of is and tall and its modified noun v.
- The function denoted by $\lambda uvc.I(\mathsf{love}, u, v, c)$ increases the value entries of c_{ijk} of c, for i, j, k indices of loves, its subject u and its object v.

As an example, consider the series of updates depicted in Figure 3.

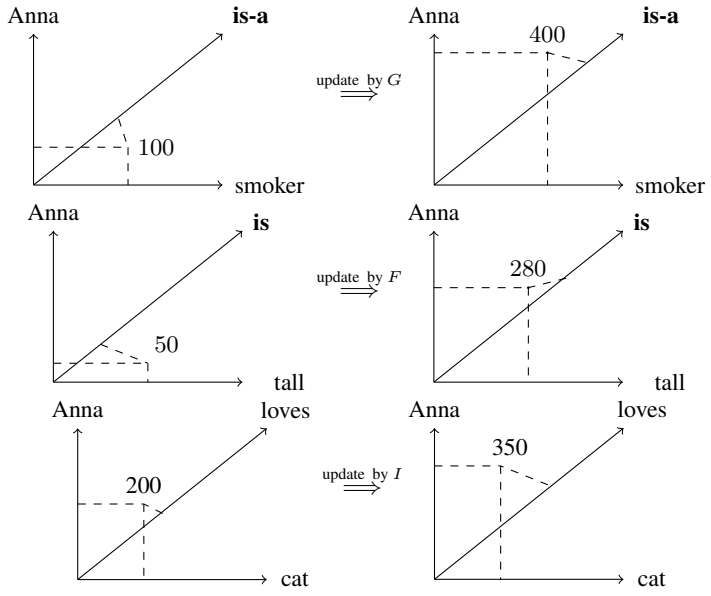

Fig. 3. An example of updates by functions F, G, I on an entity-relation cube

Relative pronouns such as 'who' update the entry corresponding to the head of the relative clause and the rest of the clause. For example in the clause 'the man who went home', we update c_{ijk} for i the index of 'man' as subject of the verb 'went' with index j and its object 'home' with index k. Propositional attitudes such as 'know' update the entry value of c_{ijk} for i the index of their subject, j the index of themselves, and j the index of their proposition. For instance in the sentence 'John knows Mary slept',

we update the entry value for 'John', 'know' and the proposition 'Mary slept'. The conjunctive *and* is modelled as before.

Negation can be modelled by assigning the abstract type $D(DS)S$ to it and a term homomorphism $\lambda uZ.\neg J(u, Z, c)$ with the object type $(V(VU))U$. Concretely, it negates the value entry of its input entity relation triple by, for example, setting $\neg J(u, Z, c) := -J(u, Z, c)$, as shown in Figure 5.

Quantifiers can be modelled in a more elaborated way. For instance *every* can have the lambda term $\lambda uZ.\forall(u, Z, c)$ assigned to it, which finds all the entries of the graph that have an is-a relation with u, then updates their value entries where they relate to the verb phrase Z by increasing them. An example is depicted in Figure 4, regarding the universal quantifier in a sentence such as "All women love cats". Here, it finds all entities that are in an is-a relation with *woman*, such as "Anna is-a woman", "Susan is-a woman", "Aunt is-a woman". Then it increases c_{ijk} where i is the index of "Anna", j is the index of "love" and k is the index of "cat", and similarly for "Susan" and "Aunt". The quantifier *some*, acts in a similar way with the difference that it only updates some of these value entries, e.g. by random choice. For example, in the above case, it only increases the entry corresponding to 'Anna' and leaves 'Aunt" and "Susan" unchanged. Generalised quantifiers act in similar ways, e.g. *most* changes most of its corresponding entries, *at least 5* changes at least 5 of them; *at most 5* changes at most 5 of them (all chosen randomly) and so on.

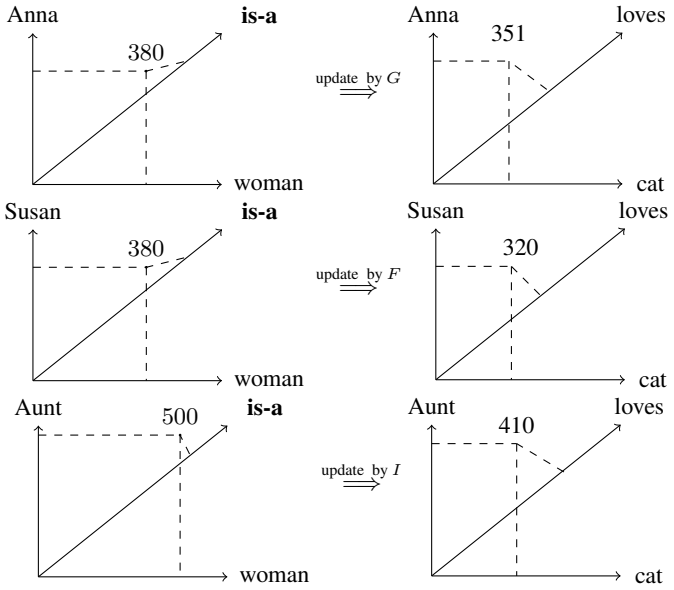

Fig. 4. An example of the updates incurred by the universal quantifier

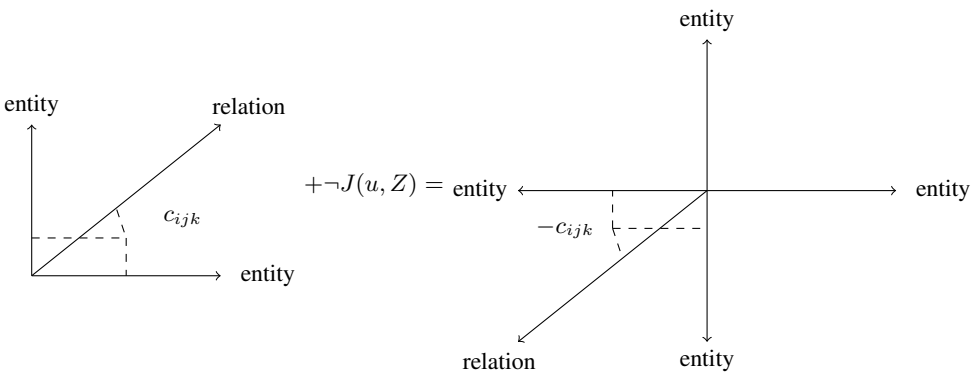

Fig. 5. The cube of an example negation operator

These update instructions fall through the semantics of phrases and sentences compositionally as in the case of co-occurrence matrices.

7 A Logic for Context Change Potentials

A logic for sentences as context change potentials has a syntax as follows:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid$$

with disjunction and implication operations defined using de Morgan duality:

$$\phi \vee \psi := \neg(\neg\phi \wedge \neg\psi)$$
$$\phi \rightarrow \psi := \neg\phi \vee \psi$$

This logic is the propositional fragment of the logic of context change potentials, presented in [20], based on the ideas of Heim [11]. Heim extends the theory of presuppositions of Karttunen [14] and defines the context change potential of a sentence as a function of the context change potentials of its parts, an idea that leads to the development of the above logic. The logic we consider here is the same logic but without the presupposition operation.

We refer to the language of this logic as $\mathcal{L}_{ccp}$. For a context c, a context change potential is defined as follows:

$$||p||(c) := c + ||p||$$
$$||\neg\phi||(c) := c - ||\phi||(c)$$
$$||\phi \wedge \psi|| := ||\psi||(||\phi||(c))$$

It is easy to verify that:

$$||\phi \vee \psi|| = ||\psi||(c) - ||\psi||(||\phi||(c))$$
$$||\phi \rightarrow \psi||(c) = c - (||\phi||(c) - ||\psi||(||\phi||(c)))$$

Here, $||\phi||$ is the context change potential of ϕ and a function from contexts to contexts. Whereas for Heim, contexts and context change potentials of atomic sentences $||p||$ are both sets of valuations, for us contexts are co-occurrence matrices or entity relation cubes and context change potentials of atomic sentences are vectors. Thus, where the context change potential operation of Heim simply takes the intersection of a context and a context change potential $c \cap ||p||$, we have to do an operation that acts on matrices/cubes rather than sets. We use the update operation of term homomorphisms, defined in the previous sections, and define a context change potential as follows:

Definition 1. *For S a sentence in $\mathcal{L}_{ccp}$, $||S||$ its context change potential, $H(S)$ the term homomorphic image of S, and c a co-occurrence matrix or an entity relation cube, we define:*

$$||S||(c) := c +' H(S)$$
$$c - H(S) := (c +' H(S))^{-1}$$

for $+'$ the update operation defined on term homomorphisms and $-'$ its inverse, defined as follows for matrices:

$$\begin{pmatrix} m_{11} & \cdots & m_{1k} \\ m_{21} & \cdots & m_{2k} \\ \vdots & & \\ m_{n1} & \cdots & m_{nk} \end{pmatrix} +' \langle a, u, v, \cdots \rangle = \begin{pmatrix} m'_{11} & \cdots & m'_{1k} \\ m'_{21} & \cdots & m'_{2k} \\ \vdots & & \\ m'_{n1} & \cdots & m'_{nk} \end{pmatrix} \quad \textit{for} \quad m'_{ij} := \begin{cases} 1 & m_{ij} = 1 \\ 1 & m_{ij} = 0 \end{cases}$$

$$\begin{pmatrix} m_{11} & \cdots & m_{1k} \\ m_{21} & \cdots & m_{2k} \\ \vdots & & \\ m_{n1} & \cdots & m_{nk} \end{pmatrix} -' \langle a, u, v, \cdots \rangle = \begin{pmatrix} m'_{11} & \cdots & m'_{1k} \\ m'_{21} & \cdots & m'_{2k} \\ \vdots & & \\ m'_{n1} & \cdots & m'_{nk} \end{pmatrix} \quad \textit{for} \quad m'_{ij} := \begin{cases} 0 & m_{ij} = 1 \\ 0 & m_{ij} = 0 \end{cases}$$

The definition of $+'$ and $-'$ for cubes are done similarly.

The $+'$ operation updates the co-occurrence matrix in a binary fashion: if the entry m_{ij} of the matrix has already been updated, and thus has value 1, then a succeeding update is not going to increase the value from 1 to 2 and will keep it as 1. Conversely, when the $-'$ operation acts on an entry m_{ij} which is already 0, it is not going to change its value, but if it acts on a non-zero m_{ij}, that is an m_{ij} which has value 1, it will decrease it to 0. The procedure is similar for the cubes. The resulting matrices and cubes will have binary entires, that is, they will either be 1's or 0's. A 1 indicates that at least one occurrence of the roles associated to the entires have been seen in the corpus before; a 0 indicates that none has been seen or that a role and its negation have occurred.

Fixing a bijection between the elements $[1, n] \times [1, k]$ of our matrices and natural numbers $[1, n \times k]$ and between elements $[1, n] \times [1, k] \times [1, z]$ of the cubes and natural

numbers $[1, n \times k \times z]$, one can show that $c +' H(S)$ is the table of a binary relation in the case of matrices and ternary relation in the case of cubes. Those entries (i, j) of the matrices and (i, j, k) of the cubes that have a non zero value entry, are mapped to an element of the relation. An example of this isomorphism is as shown below for a 2 $\times$ 2 matrix:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \mapsto \quad \mapsto \quad \begin{array}{c|cc} & 1 & 2 \\ \hline 1 & 1 & 0 \\ 2 & 1 & 1 \end{array} \qquad \{(1,1),(2,1),(2,2)\}$$

These binary updates can be seen as providing a notion of 'contextual truth', that is, for example, a sentence S is true in a given a context c, whenever the update resulting from s is already included in the matrix or cube of its context, i.e. its update is one that does not change c.

As argued in [20], the semantics of this logic is dynamic, in the sense that the context change potential of a sequence of sentences is obtained by function composition, that is as follows:

$$||S_1, \cdots, S_n||(c) := ||S_1|| \circ \cdots \circ ||S_n||(c)$$

Using this dynamic semantics, it is straightforward to show that

Proposition 1. *The context c corresponding to the sequence of sentences $S_1, \cdots, S_n$, is the zero vector updated by that sequence of sentences, that is:*

$$c = ||S_1, \cdots, S_n||(0)$$

where

$$||S||(c) := c + H(S)$$
$$c - H(S) := (c + H(S))^{-1}$$

In the case of the co-occurrence matrices, c is the co-occurrence matrix and 0 is the zero matrix. In the entity relation cube case, c is the entity relation cube and 0 is the zero cube. We are using the usual real number addition and subtraction on m_{ij} and c_{ijk} entires of the matrices and cubes, that is

$$m'_{ij} := m_{ij} + 1 \qquad m'_{ij} := m_{ij} - 1$$
$$c'_{ijk} := c_{ijk} + 1 \qquad c'_{ijk} := c_{ijk} - 1$$

We will refer to a sequence of sentences as a *corpus*.

8 Admittance of Sentences by Contexts

A notion of *admittance of a sentence by a context* was developed by Karttunen for presuppositions and extended by Heim for context change potentials. It is defined as follows, for c a context and ϕ a proposition of $\mathcal{L}_{ccp}$:

$$\text{context } c \text{ admits proposition } \phi \iff ||\phi||(c) = c$$

We use this notion and develop a similar notion between a corpus and a sentence.

Definition 2. *A corpus admits a sentence iff the context* c *(a co-occurrence matrix or entity relation cube) built from it, admits it.*

Consider the following corpus:

'Cats and dogs are animals that sleep. Cats chase cats and mice. Dogs chase all animals. Cats like mice, but mice fear cats, since cats eat mice. Cats smell mice and mice run from cats.'

It admits the following sentences:

Cats are animals.
Dogs are animals.
Cats chase cats.
Cats chase mice.
Dogs chase cats and dogs.

Note that this notion of admittance caters for monotonicity of inference. For instance, in the above example, from the sentences "Cats [and dogs] are animals [that sleep]" and "Dogs chase all animals", we can infer that the context admits the sentence "Dogs chase cats".

On the other hand, c does not admit the negation of the above, for example it does not admit

(*) Dogs do not chase cats.
(*) Dogs do not chase dogs.

It also do not admit the negations of derivations of the above or negations of sentences of the cropus, for example, it does not admit

(*) Cats are not animals.
(*) Dogs do not sleep.

The corpus misses a sentence asserting that mice are also animals. Hence, c does not admit the sentence 'dogs chase mice'. Some other sentences that are not admitted by c as as follows:

(*) Cats like dogs.
(*) Cats eat dogs.
(*) Dogs run from cats.
(*) Dogs like mice.
(*) Mice fear dogs.
(*) Dogs eat mice.

One can argue that by binarizing the update operation and using $+'$ and $-'$ rather than the original $+$ and $-$, we are loosing the full power of distributional semantics. It seems wasteful to rather than building context matrices by counting co-occurrences, only record if something co-occurred with another or not. This can be overcome by working with a pair of contexts: a binarized one and a numerical one. The binarized

	1 animal	2 sleep	3 chase	4 like	5 fear	6 eat	7 smell	8 run
1- cats	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
2- mice	0	0	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
3- dogs	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0

Table 6. The normalised co-occurrence matrix built from the example corpus with the co-occurrence window taken to be occurrence within the same sentence.

context allows for defining a notion of admittance as before, and the numerical one allows to use numerical values, e.g. the degrees of similarity between words. The notion of word similarity used in distributional semantics is a direct consequence of the distributional hypothesis: it says that words that often occur in the same contexts have similar meanings [7]. Various formal notions have been used to measure the above degree of similarity, amongst the successful ones is the cosine of the angle between the vectors of the words. If the vectors are normalised to have length 1, which we shall assume, cosine becomes the same as the dot product of the vectors. Then, one can use these degrees of similarity to assign a numerical value to the admittance relation, e.g. as follows:

> A pair of binary and numerical co-occurrence matrices c and c'admit a sentence s' with degree d, if c admits s, and s' is obtained from s by replacing a word w of s with a word w' such that w' has the same grammatical role in s' as w in s and the degree of similarity between w and w' is d, computed from the numerical entries of c'.

Here, c admits s and if there is a word in s that is similar to another word w', then if we replace w in s with w' (keeping the grammatical role that w had in s) then the sentence resulting from this substitution, i.e. s' is also admitted by c, albeit with a degree equal to the degree of similarity between w and w'. This degree is computed using the numerical values recored in c'. The above can be extended to the case when one replaces more than one word in s with words similar to them. Then, the degree of entailment may be obtained by multiplying the degrees of similarities of the individually replaced words.

The normalised context matrix of our example corpus above are as in tables 6, where for simplicity the co-occurrence window is taken to be "occurrence within the same sentence".

From the context matrix one obtains the following degrees of similarity:

$$\cos(\text{cats}, \text{mice}) = 6 \times (\frac{1}{6} \times \frac{1}{8}) = \frac{1}{8}$$
$$\cos(\text{cats}, \text{dogs}) = (\frac{1}{2}) \times 2 \times (\frac{1}{4} \times \frac{1}{8}) = \frac{1}{32}$$
$$\cos(\text{dogs}, \text{mice}) = \frac{1}{4} \times \frac{1}{6} = \frac{1}{24}$$

The corpus misses an explicit sentence declaring that mice are also animals. Hence, from the sentences of the corpus the negation of 'dogs chase mice' follows, which is a wrong entailment in the real world. This wrong can now be put right, since we can replace the word 'Cats' in the admitted sentence 'Cats are animals' with 'Mice'; as we have $\cos(\text{cats}, \text{mice}) = \frac{1}{8}$, thus obtaining the situation where c admits the following, both with degree $\frac{1}{8}$:

Mice are animals.
Dogs chase mice.

These were not possible before. We also obtain admittance of the following sentences albeit with a lower degree of $\frac{1}{24}$:

(*) Cats like dogs.
(*) Cats eat dogs.
(*) Dogs run from cats.

Some other examples are as follows with a still lower degree of $\frac{1}{32}$:

(*) Dogs like mice.
(*) Mice fear dogs.
(*) Dogs eat mice.

Some of the above are as likely as the ones that were derived with degree $\frac{1}{8}$. This is since the degrees come from co-occurrences in corpora and the one that we have is quite restrictive. One hopes that the bigger a corpus, the more reflective of the real world it is. Another way of improving the word-based entailments is by using linguistic resources such as WordNet, e.g. replacing words with their hypernyms.

8.1 Evaluating on Existing Entailment Datasets

It remains to show if the notion of admittance of a sentence by a context can be applied to derive entailment relations between sentences. In future work, we will put this method to test on the main and down stream inference datasets such as FraCas [6], SNLI[3], Zeichner[32] and datasets of the RTE challenge. The FraCas inferences are logical and the lambda calculus models of language should help in deriving them. As an example, consider the `fracas-013` test case:

`fracas-013` answer: yes
$P1$ Both leading tenors are excellent.
$P2$ Leading tenors who are excellent are indispensable.
Q Are both leading tenors indispensable?
H Both leading tenors are indispensable.

In our setting, using the updates resulting from P1 and P2, one can contextually derive H. Zeichner however does take the similarity between words into account. An example is the following entailment between two sentences; this entailment was judged to be true with confidence by human annotators:

Parents have a great influence on the career development of their children.
Parents have a powerful influence on the career development of their children.

We can derive the above with a contextual entailment consisting of a cube updated by just the above two sentences, with the degree of similarity between 'powerful' and 'great', mined from the co-occurrence matrix of a large corpus.

The judgements of the SNLI dataset are more tricky as they rely on external knowledge. For example consider the entailment between the following sentences:

A soccer game with multiple males playing.
Some men are playing a sport.

or the contradiction between the following:

A black race car starts up in front of a crowd of people.
A man is driving down a lonely road.

Deciding these correctly is a challenge for our framework. The strength of our approach is in deciding whether a set of sentences follow from a given corpus of text, rather than in judging entailment relations between a given pair or triple of sentences. We shall, nevertheless, try to experiments with all these datasets.

9 Conclusion and Future Directions

We showed how a static interpretation of a lambda calculus model of natural language provides vector representations for phrases and sentences. Here, the type of the vector of a word depended on its abstract type and could be an atomic vector, a matrix, or a cube, or a tensor of higher rank. Combinations of these vary based on the tensor rank of the type of each word involved in the combination. For instance, one could take the matrix multiplication of the matrix of an intransitive verb with the vector of its subject, whereas for a transitive verb the sequence of operations were a contraction between the cube of the verb and the vector of its object followed by a matrix multiplication between the resulting matrix and the vector of the subject. A toolkit of functions needed to perform these operations was defined. This toolkit can be restated for the higher order types, such I^2R and I^3R, rather than the current IR, to provide means of combining matrices, cubes, and their updates, if needed.

We extended the above setting by reasoning about the notion of context and its update and developing a dynamic vector interpretation for the language of lambda terms. Truth conditional and vector models of language follow two very different philosophies. The vector models are based on contexts, the truth models on denotations. Our first interpretation was static and based on truth conditions. Our second approach is based on a dynamic interpretation, where we followed the context update model of Heim, and hence, is deemed the more appropriate choice. We showed how Heim's files can be turned into vector contexts and how her context change potentials can be used to provide vector interpretations for phrases and sentences. We treated sentences as Heim's 'context change potentials' and provided update instructions for words therein–including

quantifiers, negation, and coordination words. We provided two concrete realisations of contexts: co-occurrence matrices and entity relation cubes and in each case detailed how these context update instructions allow contexts thread through vector semantics in a compositional manner. With an eye towards a large scale empirical evaluation of the model, we defined a notion of 'contexts admitting sentences' and degrees thereof between contexts and sentences and showed, by means of examples, how these can be used to judge whether a sentence is entailed by a cube context or a pair of cube and matrix contexts. A large scale empirical evaluation of the model constitutes work in progress.

Our approach is applicable to the lambda terms obtained via other syntactic models, e.g. CCG, and Lambek Grammars and can also be modified to develop a vector semantics for LFG. We also aim to work with other update semantics, such as continuation-based approaches. One could also have a general formalisation wherein both the static approach of previous work and the dynamic one of this work cohabit. This can be done by working out a second pair of type-term homomorphisms that will also work with Heim's possible world part of the contexts. In this setting, the two concepts of meaning: truth theoretic and contextual, each with its own uses and possibilities, can work in tandem.

An intuitive connection to Fuzzy logic is imaginable, wherein one interprets the logical words in more sophisticated ways, for instance, conjunction and disjunction take max and min of their entires, or add and subtract them. It may be worth investigating if such connections add to the applicability of the current model and if so making the connection formal.

Acknowledgements We wish to thank the anonymous referees of a short version of this paper presented in LACL 2017 for excellent feedback. The research done for this paper was supported by the Royal Society International Exchange Award IE161631.

References

1. Baroni, M., Zamparelli, R.: Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In: Conference on Empirical Methods in Natural Language Processing (EMNLP-10). Cambridge, MA (2010)
2. Baroni, M., Bernardi, R., Zamparelli, R.: Frege in space: A program for compositional distributional semantics. Linguistic Issues in Language Technology 9, 5–110 (2014)
3. Bowman, S., Angeli, G., Potts, C., Manning, C.: A large annotated corpus for learning natural language inference. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics (2015)
4. Coecke, B., Sadrzadeh, M., Clark, S.: Mathematical Foundations for Distributed Compositional Model of Meaning. Lambek Festschrift. Linguistic Analysis 36, 345–384 (2010)
5. Coecke, B., Grefenstette, E., Sadrzadeh, M.: Lambek vs. lambek: Functorial vector space semantics and string diagrams for lambek calculus. Ann. Pure Appl. Logic 164(11), 1079–1100 (2013)
6. Cooper, R., Crouch, D., Van Eijck, J., Fox, C., Van Genabith, J., Jaspars, J., Kamp, H., Milward, D., Pinkal, M., Poesio, M., et al.: Using the framework. Tech. rep., Technical Report LRE 62-051 D-16, The FraCaS Consortium (1996)

7. Firth, J.R.: A Synopsis of Linguistic Theory, 1930-1955. Studies in Linguistic Analysis pp. 1–32 (1957)
8. Grefenstette, E., Sadrzadeh, M.: Concrete models and empirical evaluations for the categorical compositional distributional model of meaning. Computational Linguistics 41, 71–118 (2015)
9. de Groote, P.: Towards a montagovian account of dynamics. Semantics and Linguistic Theory 16, 1–16 (2006)
10. de Groote, P.: Towards Abstract Categorial Grammars. In: Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference. pp. 148–155. ACL, Toulouse, France (2001)
11. Heim, I.: On the projection problem for presuppositions. In: Portner, P., Partee, B.H. (eds.) Formal Semantics - the Essential Readings, pp. 249–260. Blackwell (1983)
12. Heim, I., Kratzer, A.: Semantics in generative grammar. Blackwell textbooks in linguistics, Blackwell publishers, Cambridge (Mass.), Oxford (1998), http://opac.inria.fr/record=b1080204
13. Kambhatla, N.: Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In: Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions. ACLdemo '04, Association for Computational Linguistics, Stroudsburg, PA, USA (2004), http://dx.doi.org/10.3115/1219044.1219066
14. Karttunen, L.: Presupposition and Linguistic Context. Theoretical Linguistics 1(1/2), 182–194 (1974)
15. Krishnamurthy, J., Mitchell, T.M.: Vector space semantic parsing: A framework for compositional vector space models. In: Proceedings of the 2013 ACL Workshop on Continuous Vector Space Models and their Compositionality (2013)
16. Lewis, M., Steedman, M.: Combined Distributional and Logical Semantics. Transactions of the Association for Computational Linguistics 1, 179–192 (May 2013), http://aclweb.org/anthology//Q/Q13/Q13-1015.pdf
17. Maillard, J., Clark, S., Grefenstette, E.: A type-driven tensor-based semantics for ccg. EACL 2014 Type Theory and Natural Language Semantics Workshop (2014)
18. Mitchell, J., Lapata, M.: Composition in distributional models of semantics. Cognitive Science 34, 1388–1439 (2010)
19. Montague, R.: The Proper Treatment of Quantification in Ordinary English. In: Thomason, R. (ed.) Formal Philosophy. Selected Papers of Richard Montague, pp. 247–270. Yale University Press (1974)
20. Muskens, R., van Benthem, J., Visser, A.: Dynamics. pp. 589–643. Elsevier (2011)
21. Muskens, R.A.: Categorial Grammar and Lexical-Functional Grammar. In: Butt, M., King, T.H. (eds.) Proceedings of the LFG01 Conference, University of Hong Kong. pp. 259–279. CSLI Publications, Stanford CA (2001), http://cslipublications.stanford.edu/LFG/6/lfg01.html
22. Muskens, R.A.: Language, Lambdas, and Logic. In: Kruijff, G.J., Oehrle, R. (eds.) Resource Sensitivity in Binding and Anaphora, pp. 23–54. Studies in Linguistics and Philosophy, Kluwer (2003)
23. Muskens, R.: New Directions in Type-Theoretic Grammars. Journal of Logic, Language and Information 19(2), 129–136 (2010)
24. Muskens, R., Sadrzadeh, M.: Context update for lamdas and vectors. In: LNCS Proceedings of the 9th International Conference on Logical Aspects of Computational Linguistics. Springer, Nancy (December 2016), to appear
25. Muskens, R., Sadrzadeh, M.: Lamdas and vectors. Workshop on Distributional Semantics and Linguistic Theory (DSALT), 28th European Summer School in Logic, Language and Information (ESSLLI) (August 2016), free University of Bozen-Bolzano

26. Poon, H., Domingos, P.: Unsupervised semantic parsing. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1. pp. 1–10. EMNLP '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009), http://dl.acm.org/citation.cfm?id=1699510.1699512
27. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III. pp. 148–163. ECML PKDD'10, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1889788.1889799
28. Rubenstein, H., Goodenough, J.: Contextual Correlates of Synonymy. Communications of the ACM 8(10), 627–633 (1965)
29. Tarski, A.: Introduction to Logic and to the Methodology of Deductive Sciences. Dover Publications (1946)
30. Veltman, F.: Defaults in update semantics. Journal of Philosophical Logic 25(3), 221–261 (1996)
31. Yao, L., Riedel, S., McCallum, A.: Unsupervised relation discovery with sense disambiguation. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1. pp. 712–720. ACL '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012)
32. Zeichner, N., Berant, J., Dagan, I.: Crowdsourcing inference-rule evaluation. In: Proceedings of ACL (short papers) (2012)