Static and dynamic vector semantics for lambda calculus models of natural language

Mehrnoosh Sadrzadeh[1] *and Reinhard Muskens*[2]
[1] School of Electronic Engineering and Computer Science,
Queen Mary University of London.
[2] Department of Philosophy, Tilburg University

ABSTRACT

Vector models of language are based on contextual aspects of language – distributions of words and how they co-occur in text. Truth conditional models focus on logical aspects of language and on how words combine to contribute to these aspects. In the truth conditional approach, there is a focus on the denotations of phrases. In vector models, the degree of co-occurrence of words in context determines how similar their meanings are. The two approaches have complementary virtues. In this paper we combine them and develop a vector semantics for language, based on the typed lambda calculus. We provide two types of vector semantics: a static one using techniques from the truth conditional tradition, and a dynamic one with a form of interpretation inspired by Heim's context change potentials. We show, with examples, how the dynamic model can be applied to entailment between a corpus and a sentence.

Keywords: simply typed lambda calculus, vector semantics, composition, context update potential, dynamic logic

1 INTRODUCTION

Vector semantic models, otherwise known as distributional models, are based on the contextual aspects of language, i.e. 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. They typically describe how words can be represented by

functions over sets, and how these functions can be composed. 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 based on a certain type of grammatical representation, e.g. the pregroup model (Coecke *et al.* 2010), the Lambek Calculus model (Coecke *et al.* 2013), and the combinatorial categorial models (Krishnamurthy and Mitchell 2013; Maillard *et al.* 2014). We also have more concrete models that draw inspiration from type theory, but whose major contribution lies in developing concrete ways of constructing linear and multi-linear algebraic counterparts for syntactic types, e.g. matrices and tensors (Grefenstette and Sadrzadeh 2015; Baroni *et al.* 2014), and relational clusters (Lewis and Steedman 2013).

What some of these approaches (Coecke *et al.* 2010; Krishnamurthy and Mitchell 2013; Maillard *et al.* 2014) lack more than others (Baroni *et al.* 2014; Lewis and Steedman 2013) is acknowledgement of the inherent gap between contextual and truth conditional semantics: they closely follow truth theoretic conditions to assign vector representations to (readings of) phrases and sentences.[1] 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 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 also at modelling entailment. It is quite conceivable that a theory combining the two as separate modules will be simpler than trying to make one of the approaches do things it was never intended for.

In this paper, we first sketch how an approach to semantics, derived in many of its aspects from that pioneered by Montague (1974), can be used to assign vector meanings to linguistic phrases. The theory will be based on simply typed lambda calculus and, as a result, will

[1] Below, when we refer to phrases and sentences, strictly speaking, we mean *readings* of phrases and sentences.

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.[2] Our reason for using 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 a semantics similar to those of the static approaches listed above. The reason for providing it is to show that a lambda calculus model of language can be directly provided with a straightforward vector semantics. As will be seen, abstract lambda terms, which can be used as translations of linguistic expressions, have much in common with the Logical Forms of these expressions, and the lambda binders in them facilitate the treatment of 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 process, and examples where the direct use of lambdas is an improvement on the above-listed static approaches.

The above semantics does not have an explicit notion of context, however. The second contribution of this 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 (1983). We show how to assign such a vector interpretation to words, and how these interpretations combine so that the vectors of the sentences containing them change the context, in a dynamic style similar to that proposed by Heim. As 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,

[2] Linguistic trees, for example, can be associated with the abstract lambda terms considered below via type-driven translation (Partee 1986; Klein and Sag 1985; Heim and Kratzer 1998). But other syntactic structures can be provided with them as well. In the framework of Lexical-Functional Grammar, abstract lambda terms can be assigned to f-structures with the help of linear logic acting as 'glue' (Dalrymple *et al.* 1993). In Combinatory Categorial Grammar, derivations can be associated with abstract lambda terms using combinators (Steedman 2000), while proofs in Lambek Categorial Grammar can be provided with them by the Curry-Howard morphism (van Benthem 1986).

encoded here in the form of cubes. Both of these are built from corpora of documents and record co-occurrence between words: in a simple neighbourhood window, in 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 for other distributional notions of contexts, such as networks of grammatical dependencies. We show how our approach relates to Heim's original notion of 'files' as contexts. Other dynamic approaches, such as update semantics (Veltman 1996) and continuation-based semantics (de Groote 2006), can also be used; we aim to do this in the future.

Compositional vector semantics is our goal, but the nature of this paper is theoretical. So we shall not propose – from the armchair so to speak – 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 leave 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 (1983) and Karttunen (1974) 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 words. As this notion resembles that of "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 applied to test this notion, but leave experimental evaluation to future work.

The lambda calculus approach we use is based on Lambda Grammars (Muskens 2001, 2003), which were independently introduced as Abstract Categorial Grammars (ACGs) in de Groote (2001). The theory developed here, however, can be based on any syntax-semantics interface that works with a lambda calculus semantics – our approach is agnostic as to the choice of a syntactic theory.

This paper is the journal version of our previous short paper (Muskens and Sadrzadeh 2016a) and extended abstract (Muskens and Sadrzadeh 2016b).

Lambda Grammars (Muskens 2001, 2003) were independently introduced as Abstract Categorial Grammars (ACGs, de Groote 2001). 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 $\mathscr{B}$ is some set of basic types, we write $TYP(\mathscr{B})$ for the smallest set containing $\mathscr{B}$ such that $(\alpha\beta) \in TYP(\mathscr{B})$ whenever $\alpha, \beta \in TYP(\mathscr{B})$. Let $\mathscr{B}_1$ and $\mathscr{B}_2$ be sets of basic types. A function η from $TYP(\mathscr{B}_1)$ to $TYP(\mathscr{B}_2)$ is said to be a *type homomorphism* if $\eta(\alpha\beta) = (\eta(\alpha)\eta(\beta))$, for all $\alpha, \beta \in TYP(\mathscr{B}_1)$. It is clear that a type homomorphism η with domain $TYP(\mathscr{B})$ is completely determined by the values of η for types $\alpha \in \mathscr{B}$.

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 $\mathscr{B}_1 = \{D, N, S\}$ (D stands for determiner phrases, N for nominal phrases, S for sentences), let $\mathscr{B}_2 = \{e, s, t\}$ (e is for entities, s for worlds, and t for truth-values), and let h_0 be defined by: $h_0(D) = e$, $h_0(N) = est$,[3] 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. Additional information about the conventions used in Table 1 is given in a footnote.[4]

We now define the notion of *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 $TYP(\mathscr{B}_1)$ and let C_2 be a set of constants typed by types from $TYP(\mathscr{B}_2)$. A function $\vartheta : \Lambda(C_1) \to \Lambda(C_2)$ is a *term homomorphism based on* η if $\eta : TYP(\mathscr{B}_1) \to TYP(\mathscr{B}_2)$ is a type homomorphism and, whenever $M \in \Lambda(C_1)$:

[3] 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*.

[4] In Table 1, p is a variable of type st, while x is of type e. The variables 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.

Table 1: An Abstract Categorial Grammar / Lambda Grammar connecting abstract terms with Montague-like meanings

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`	$N(DS)S$	$\lambda P'\lambda P\lambda w.\exists x(P'xw \wedge Pxw)$	$(est)(est)st$

- $\vartheta(M)$ 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 mechanism can be used, let us continue with the same example. 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 column. 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 under H_0. For example, H_0 sends the abstract term:[5]

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

[5] 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, to improve clarity, we will bend the rules and write $M(N_1,\ldots,N_n)$ for $(MN_1\ldots N_n)$ or $A \wedge B$ for $\wedge AB$, for example.

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

$$\lambda w \exists y(\mathit{woman}\, y w \wedge \forall x(\mathit{man}\, x w \rightarrow \mathit{love}\, y x w)) \; .$$

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

This example sends abstract terms to translations that are close to those of Montague (1974). While such translations obviously will not serve as *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 de Groote (2001) and Muskens (2003, 2010).

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 $\mathbb{R}$, so that constants such as the following (of the types shown) have their usual interpretation:[6]

$$\begin{array}{rcl} 0 & : & R \\ 1 & : & R \\ + & : & RRR \\ \cdot & : & RRR \\ < & : & RRt \end{array}$$

This can be done by imposing one of the sets of second-order axioms in Tarski (1965). Given these axioms, we have $D_R = \mathbb{R}$ in full models, whereas we have non-standard models under the Henkin interpretation (Henkin 1950).

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

[6] Constants such as $+$, $\cdot$, and $<$ will be written between their arguments.

Table 2: Vector types and their abbreviations

Type	Math Abbreviation	Letter Abbreviation	Description
IR	(I^1R)	V	Vector
IIR	I^2R	M	Matrix
$IIIR$	I^3R	C	Cube
$\vdots$	$\vdots$		

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). In this paper, we work with a single index type, but one may 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. In the following definitions, r is of type R; i, j, and k are of type I; v and u are of type V; m and c are of types M and C respectively; and indices are written as subscripts, so 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 various *rotation* operations with type $\rho : VVV$.

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 different way from that used in Section 2. A very simple type homomorphism h can be defined by:

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

So h assigns vectors to determiners, nominal phrases, and sentences. There are other possibilities for the range of h and, in the following section, we will sketch a more elaborate assignment in which a running context is used. 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. Here, Z is a variable of type VV, and v and u are of type V. As an example, consider the constant `woman`; it 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:[7]

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

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

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

In Table 3, nominal phrases like woman are represented by vectors, 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. There is still much debate as to

[7] 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 easily be guessed.

Table 3: A fragment of static vector semantics. Abstract constants c are typed with abstract types τ and their term homomorphic images $H(c)$ typed by $h(\tau)$

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

Table 4: Term homomorphic images $H(c)$ for pointwise addition and multiplication, and matrix multiplication as composition operations

Addition	Multiplication	Matrix Multiplication
$H(c)$	H(c)	H(c)
woman	woman	woman
$\lambda v.(\text{tall} \boxplus v)$	$\lambda v.(\text{tall} \odot v)$	$\lambda v.(\text{tall} \times_1 v)$
$\lambda v.(\text{smoke} \boxplus v)$	$\lambda v.(\text{smoke} \odot v)$	$\lambda v.(\text{smoke} \times_1 v)$
$\lambda uv.(\text{love} \boxplus u) \boxplus v$	$\lambda uv.(\text{love} \odot u) \odot v$	$\lambda uv.(\text{love} \times_2 u) \times_1 v$
$\lambda uv.(\text{know} \boxplus u) \boxplus v$	$\lambda uv.(\text{know} \odot u) \odot v$	$\lambda uv.(\text{know} \times_2 u) \times_1 v$
$\lambda vZ.Z(\text{every} \boxplus v)$	$\lambda vZ.Z(\text{every} \odot v)$	$\lambda vZ.Z(\text{every} \times_1 v)$
$\lambda vZ.Z(\text{a} \boxplus v)$	$\lambda vZ.Z(\text{a} \odot v)$	$\lambda vZ.Z(\text{a} \times_1 v)$

the best operations for composing vectors. Mitchell and Lapata (2010) consider pointwise addition and multiplication of vectors, while matrix multiplication is used in Baroni and Zamparelli (2010). 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 other columns. The $H(c)$ columns for these models are given in Table 4.[8]

In this paper, we will not choose between these operations. Instead, we will explore how to combine such functions once an initial set has been established (and validated empirically). Functions in the initial set will typically combine vector meanings of adjacent phrases. Like Baroni *et al.* (2014), who provide an excellent introduction to and review of compositional vector semantics, our aim has been to pro-

[8] In Table 4, we use the same typographical conventions for variables as in Table 3, while, in its first two alternative columns, all constants written in sans serif are taken to be of type V. In its third column, the types of these constants (and in fact the whole column) are as in Table 3 again.

pose a general theory that also includes dependencies between non-adjacent phrases, e.g., in topicalisation or relative clause formation.

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 $\mathscr{D}_I$ of individuals, as follows:

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

We follow Heim (1983) 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 – given any context, a sentence will transform it into a new context. In particular, a sentence S comes provided with a sequence of instructions that, given any context *ctx*, updates its information so that a new context results, denoted as:

$$ctx + S$$

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 labelled by words from a vocabulary Σ, and with entries taking values from $\mathbb{R}$ (for a full description see Rubenstein and Goodenough 1965). Thus, M can be seen as the set of its vectors:

$$\{\overrightarrow{v} \mid \overrightarrow{v} : \Sigma \to \mathbb{R}\},$$

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

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

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

Thus a distributional context matrix is obtainable by endowing Heim's contexts with $\mathbb{R}$. In other words, we are assuming not only that 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 distributional semantics is arguable, as vectors retrieved from a corpus are not naturally truth conditional. Keeping the possible worlds in the picture provides a mechanism to assign a proposition to a distributional vector by other means and can become very useful. We leave working with possible worlds to future studies and in this paper only work with sets of vectors as our contexts, as follows:

$$ctx \subseteq \{\overrightarrow{g} \mid \overrightarrow{g} : \Sigma \to \mathbb{R}, g \in M\} .$$

Distributional versions of CCPs can be defined based on Heim's intuitions and definitions. In what follows, we show 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 therefore assign the type U (for 'update') to them. A context can be a matrix or a cube, so it can be of type I^2R or I^3R. A sentence can then be of 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. The notation U can abbreviate either, depending on whether we choose to model contexts as cubes or as matrices. The concrete semantics obtained by each choice will be discussed in more detail in Section 5 and Section 6, respectively.

Update functions are presented in Table 5, 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 following paragraphs. In the schematic entry for and, we write $\rho(\overline{\alpha})$ for $\rho(\alpha_1)\cdots\rho(\alpha_n)$, if $\overline{\alpha} = \alpha_1\cdots\alpha_n$. 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 these 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 of 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 higher type, i.e. $((VU)U)((VU)U)$, via the functions Z and Q, respectively.

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: A fragment of dynamic vector semantics. Abstract constants a typed with abstract types τ and their term homomorphic images $H(a)$ typed by $\rho(\tau)$

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 takes a vector for an intransitive verb, e.g. smoke, a vector v for its subject, plus a context c, and returns a modified context $G(\mathsf{smoke}, v, c)$. The function I takes a vector for a transitive verb, 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 similar to that of 'tall', but this would not make much sense. Content words, and not function words, seem to be important in a distributional setting.

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 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 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 pass through 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 an initial 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 ⟨tall, woman⟩ and ⟨woman, smokes⟩ are increased. Much longer chains of context updates can be 'threaded' in this way.

In the following, we give some examples. In each case, sentence a is followed by an abstract term in b, thus capturing 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. `(a stockbroker)`$\lambda\xi$`.Sue(and admires loves` ξ`)`

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. `(every cop)`
`(`$\lambda\xi$`.and(Anna(despise` ξ`))(Bill(admire` ξ`)))`

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. `the(who(`$\lambda\xi$`.Bill(claims(Anna(saw` ξ`))))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 (Rubenstein and Goodenough 1965). Given a corpus of texts, a co-occurrence matrix has, for each of its entries, the degree of co-occurrence between that word and neighbouring words. The neighbourhood is usually a window of k words on either side of the word. The update type U associated with 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-occurrence between the labelling words of the rows and columns of the matrix. In this paper, to keep things simple, we work on 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 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 an example, see 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 in the sentence. 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 in a sentence are based on the grammatical roles of those words, which act as 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 entry values, depicted below:

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

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

- The function $G(\mathsf{smoke}, v, c)$ increases the entry value of m_{ij} in c by 1 in case i is the index of smoke and j is the index of its subject v. In all other cases $m'_{ij} = m_{ij}$.
- The function $I(\mathsf{love}, u, v, c)$ increases the entry values of m_{ij}, m_{jk}, and m_{ik} in c by 1 in case i is the index of loves, j is the index of its subject u, and k the index of its object v. In all other cases $m'_{ij} = m_{ij}$.

- The function $F(\mathsf{tall}, v, c)$ increases the entry value of m_{ij} in c by 1 in case i is the index of tall and j is the index of its modified noun v. In all other cases $m'_{ij} = m_{ij}$. The entry for *tall* in Table 1 uses this function, but allows for further update of context.
- The function $J(\mathsf{know}, v, c)$ increases the entry value of m_{ij} in c by 1 in case i is the index of know and j is the index of its subject v. In all other cases $m'_{ij} = m_{ij}$. The updated matrix becomes the input for further update (by the context change potential of the sentence that is known).

As an example, consider the co-occurrence matrices depicted in Figure 1. The initial matrix is a snapshot just before a series of updates are applied. The rationale of this example is as follows: Anna is a woman and so this word is not frequently found in the context man; as a result, it has a low value of 100 at that entry; Anna loves cats (and has some herself), so the entry at the context cat is 700; she loves other things, such as smoking, and so there is a substantial

		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

a series of updates by
F, G, I, and 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

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

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 initial matrix in Figure 1, we use ϵ to indicate a predefined fixed value.

The intransitive verb `smokes` updates the initial matrix in 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, the 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` would raise this number from 300 to 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 entries in row 6, that is $c_{61}, c_{62}, c_{63}, c_{64}$ and c_{65}, will be updated by function J, for instance, to the values in the updated matrix.

6 ENTITY RELATION CUBE CONTEXT AND ITS UPDATE

A corpus of texts can be seen as a sequence of lexical items occurring in the vicinity of each other, and can thus be transformed into a co-occurrence matrix. It can also be seen as a sequence of entities

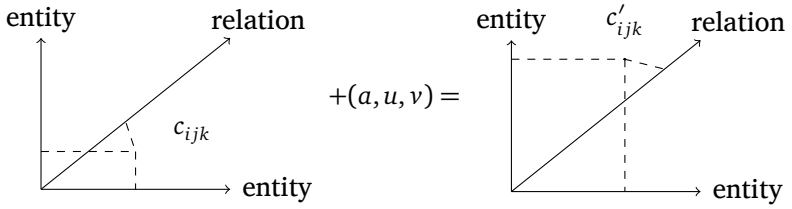

Figure 2: Updates of entries in an entity relation cube

where $c'_{ijk} := c_{ijk} + 1$

related to each other via predicate-argument structures, which can therefore be transformed into an entity relation graph. This 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 a variety of resources: a semantic network of concepts, a knowledge base such as WordNet or FrameNet. We work with entities and relations extracted from text. Creating such graphs from text corpora automatically has been the subject of much recent research (see, for example, Yao *et al.* 2012, and Riedel *et al.* 2010, for a direct approach; see also Kambhatla 2004, and Poon and Domingos 2009, for an approach based on semantic parsing). The elements of an entity relation graph are argument-relation-argument triples, sometimes referred to as *relation paths* (Yao *et al.* 2012). Similarly to Lewis and Steedman (2013), we position ourselves in a binary version of the world, where all relations are binary; we turn unary 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 entry values greater than or equal to the originals, as depicted in Figure 2.

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

- The function $G(\mathsf{smoke}, v, c)$ increases the entry value c_{ijk} of c in case i is the fixed index of is-a, j is the index of smoker, and k is the index of v, the subject of smoke. Other entry values remain unchanged.
- The function $F(\mathsf{tall}, v, c)$ increases the entry value c_{ijk} of c in case i is the fixed index of is, j is the index of tall, and k is the index of v, the modified noun. Other entry values remain unchanged.
- The function denoted $I(\mathsf{love}, u, v, c)$ increases the entry value c_{ijk} of c in case i is the index of love, j is the index of its subject u, and k is the index of its object v. Other entry values remain unchanged.

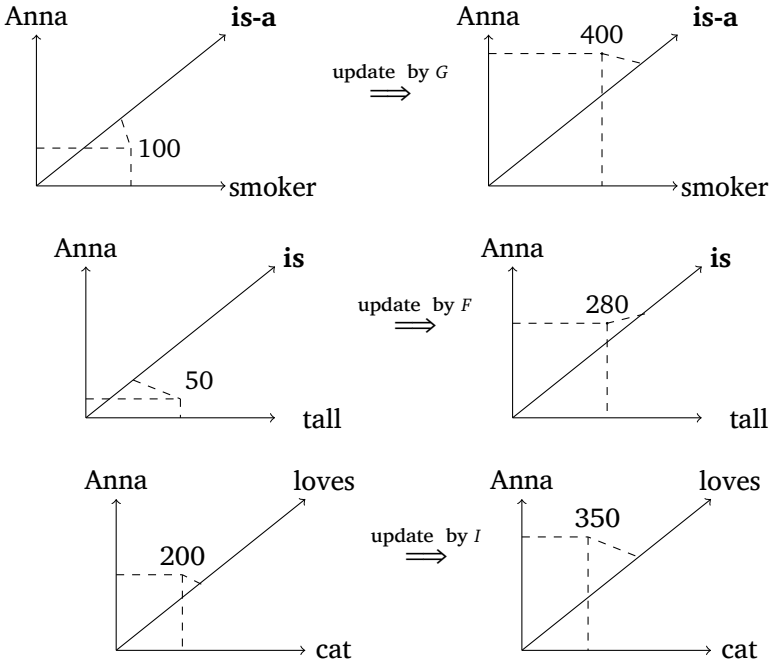

Figure 3: An example of updates by functions F, G, I on an entity-relation cube

As an example, consider the series of updates depicted in Figure 3. 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 (see also Section 4, example (3)). 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 k 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 (in Section 4, compare examples (1) and (2)).

Negation can be modelled by providing an abstract term `not` of type SS with a term homomorphic image $\lambda pc.c \div pc$ of type UU, where $\div$ is pointwise subtraction of cubes (i.e. $\lambda cc'ijk.cijk - c'ijk$). The operation denoted by this term first updates the context with the non-negated sentence, after which the result is subtracted from the context again.

7 LOGIC FOR CONTEXT CHANGE POTENTIALS

The logic for sentences as context change potentials has the following syntax.

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

Disjunction and implication operations are defined using the De Morgan duality.

$$\begin{aligned} \phi \vee \psi &:= \neg(\neg\phi \wedge \neg\psi) \\ \phi \rightarrow \psi &:= \neg\phi \vee \psi \end{aligned}$$

This logic is the propositional fragment of the logic of context change potentials, presented in Muskens *et al.* (1997), based on the ideas of Heim (1983). Heim extends Karttunen's theory of presuppositions (Karttunen 1974) 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 $\mathscr{L}_{ccp}$. For a context c, a context change potential is defined as follows.

$$\begin{aligned} \|p\|(c) &:= c + \|p\| \\ \|\neg\phi\|(c) &:= c - \|\phi\|(c) \\ \|\phi \wedge \psi\| &:= \|\psi\|(\|\phi\|(c)) \end{aligned}$$

It is easy to verify that:

$$\begin{aligned} \|\phi \vee \psi\| &= \|\psi\|(c) - \|\psi\|(\|\phi\|(c)) \\ \|\phi \rightarrow \psi\|(c) &= c - (\|\phi\|(c) - \|\psi\|(\|\phi\|(c))) \,. \end{aligned}$$

Here, $\|\phi\|$ is the context change potential of ϕ and a function from contexts to contexts. Whereas, for Heim, both contexts and context change potentials of atomic sentences $\|p\|$ are 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 (Heim 1983) simply takes the intersection of a context and a context change potential $c \cap \|p\|$, we perform 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:*

$$\begin{aligned} \|S\|(c) &:= c +' H(S) \\ c - H(S) &:= (c +' H(S))^{-1} , \end{aligned}$$

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

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

$$\textit{for} \quad m'_{ij} := \begin{cases} 1 & m_{ij} = 1 \\ 1 & m_{ij} = 0 \end{cases}$$

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

$$\textit{for} \quad m'_{ij} := \begin{cases} 0 & m_{ij} = 1 \\ 0 & m_{ij} = 0 \end{cases}$$

The definitions of $+'$ and $-'$ for cubes are similar.

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 will not increase the value from 1 to 2 but will keep it as 1. Conversely, when the $-'$ operation acts on an entry m_{ij} which is already 0, it will not 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 cubes. The resulting matrices and cubes will have binary entries, that is, they will either be 1 or 0. A 1 indicates that at least one occurrence of the roles associated with the entries has previously been seen in the corpus; 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 of a 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 entry value are mapped to an element of the relation. An example of this isomorphism is shown below for a 2×2 matrix:

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

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 Muskens *et al.* (1997), 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, as follows:

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

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:*

$$c = \|S_1,\ldots,S_n\|(0) \; ,$$

where

$$\begin{aligned} \|S\|(c) &:= c + H(S) \\ c - H(S) &:= (c + H(S))^{-1} \; . \end{aligned}$$

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

$$\begin{aligned} m'_{ij} &:= m_{ij} + 1 \qquad & m'_{ij} &:= m_{ij} - 1 \\ c'_{ijk} &:= c_{ijk} + 1 \qquad & c'_{ijk} &:= c_{ijk} - 1 \; . \end{aligned}$$

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

8 ADMITTANCE OF SENTENCES BY CONTEXTS

The notion of *admittance of a sentence by a context* was developed by Karttunen (1974) for presuppositions, and extended by Heim (1983) for context change potentials. We here define it as follows, for c a context and ϕ a proposition of $\mathscr{L}_{ccp}$.

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

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 does not admit the negations of derivations of the above or negations of sentences of the corpus, 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 are 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 losing the full power of distributional semantics. It seems wasteful simply to record the presence or absence of co-occurrence, rather than build context matrices by counting co-occurrences. This can be overcome by working with a pair of contexts: a binarized one and a numerical one. The binarized context allows a notion of admittance to be defined as before, and the numerical one allows the use of 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, where words that often occur in the same contexts have similar meanings (Firth 1957). 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. One can then 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), the sentence resulting from this substitution,

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

	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

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 recorded in c'. The above can be extended to the case where 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 similarity of the individually replaced words.

The normalised context matrix of our example corpus above is as in Table 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.

$$\begin{aligned} \cos(\text{cats}, \text{mice}) &= 6 \times \left(\frac{1}{6} \times \frac{1}{8}\right) = \frac{1}{8} \\ \cos(\text{cats}, \text{dogs}) &= \left(\frac{1}{2}\right) \times 2 \times \left(\frac{1}{4} \times \frac{1}{8}\right) = \frac{1}{32} \\ \cos(\text{dogs}, \text{mice}) &= \frac{1}{4} \times \frac{1}{6} = \frac{1}{24} \end{aligned}$$

The corpus lacks 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 those that were derived with degree $\frac{1}{8}$. This is because the degrees come from co-occurrences in corpora, and our corpus is quite limited. One hopes that the bigger the corpus, the more reflective of the real world it will be. Another way of improving 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 see 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 the test on inference datasets such as FraCaS (Cooper *et al.* 1996), SNLI (Bowman *et al.* 2015), the dataset in Zeichner *et al.* (2012), and the datasets in 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. In Zeichner *et al.* (2012), the similarity between words is also taken into account. An example is the following entailment between two sentences; this entailment was judged to be valid with confidence by human annotators:

Parents have 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.

Judgements on the SNLI dataset are more tricky, as they rely on external knowledge. For example, consider the entailment between the following phrases:

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 follows from a given corpus of texts, rather than in judging entailment relations between a given pair or triple of sentences. Nevertheless, we shall try to experiment 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 was 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 types of tensors of higher order, such as I^2R and I^3R,

rather than the current *IR*, to provide a 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. Vector models are based on contexts, 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 (1983), 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, i.e. co-occurrence matrices and entity relation cubes, and in each case detailed how these context update instructions allow contexts to 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 notions can be used to judge whether a sentence is entailed by a cube context or by a pair of cube and matrix contexts. A large-scale empirical evaluation of the model is currently underway.

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 achieved 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 entries, or add and subtract them. It may be worth investigating if such connections add to the applicability of the current model and, if so, make the connection formal.

ACKNOWLEDGEMENTS

We wish to thank the anonymous referees for their very valuable remarks. The anonymous referees of a short version of this paper, presented at LACL 2017, also gave excellent feedback. Carmela Chateau Smith's meticulous copy-editing considerably improved the readability of the paper and the grammaticality of its phrasing. All remaining errors are ours. The research for this paper was supported by the Royal Society International Exchange Award IE161631.

REFERENCES

Marco BARONI, Raffaella BERNARDI, and Roberto ZAMPARELLI (2014), Frege in space: A program for compositional distributional semantics, *Linguistic Issues in Language Technology*, 9:241–346.

Marco BARONI and Roberto ZAMPARELLI (2010), Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space, in Hang LI and Lluìs MÀRQUEZ, editors, *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pp. 1183–1193, Association for Computational Linguistics, Stroudsburg, PA, `http://aclweb.org/anthology/D10-1115`.

Johan VAN BENTHEM (1986), *Essays in Logical Semantics*, Studies in Linguistics and Philosophy 29, Reidel, Dordrecht.

Samuel BOWMAN, Gabor ANGELI, Christopher POTTS, and Christopher MANNING (2015), A large annotated corpus for learning natural language inference, in Lluìs MÀRQUEZ, Chris CALLISON-BURCH, and Jian SU, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP '15, pp. 632–642, Association for Computational Linguistics, Stroudsburg, PA, `http://aclweb.org/anthology/D15-1075`.

Bob COECKE, Edward GREFENSTETTE, and Mehrnoosh SADRZADEH (2013), Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus, *Annals of Pure and Applied Logic*, 164(11):1079–1100.

Bob COECKE, Mehrnoosh SADRZADEH, and Stephen CLARK (2010), Mathematical foundations for distributed compositional model of meaning, *Linguistic Analysis*, 36:345–384.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, and Massimo Poesio (1996), *Using the framework*, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Mary Dalrymple, John Lamping, and Vijay Saraswat (1993), LFG semantics via constraints, in *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*, EACL '93, pp. 97–105, Association for Computational Linguistics, Stroudsburg, PA, http://aclweb.org/anthology/E93-1013.

John Rupert Firth (1957), A synopsis of linguistic theory, 1930–1955, in *Studies in Linguistic Analysis*, pp. 1–32, Blackwell, Oxford.

Edward Grefenstette and Mehrnoosh Sadrzadeh (2015), Concrete models and empirical evaluations for the categorical compositional distributional model of meaning, *Computational Linguistics*, 41:71–118.

Philippe de Groote (2001), Towards abstract categorial grammars, in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and 10th Conference of the European Chapter of the Association for Computational Linguistics*, ACL '01, pp. 252–259, Association for Computational Linguistics, Stroudsburg, PA, http://aclweb.org/anthology/P01-1033.

Philippe de Groote (2006), Towards a Montagovian account of dynamics, *Semantics and Linguistic Theory*, 16:1–16.

Irene Heim (1983), On the projection problem for presuppositions, in *Proceedings of the Second Annual West Coast Conference on Formal Linguistics*, pp. 114–125, reprinted in Portner and Partee (2002).

Irene Heim and Angelika Kratzer (1998), *Semantics in Generative Grammar*, Blackwell textbooks in linguistics, Blackwell, Oxford, ISBN 0-631-19712-5.

Leon Henkin (1950), Completeness in the theory of types, *Journal of Symbolic Logic*, 15:81–91.

Nanda Kambhatla (2004), 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, http://aclweb.org/anthology/P04-3022.

Lauri Karttunen (1974), Presupposition and linguistic context, *Theoretical Linguistics*, 1(1–3):182–194.

Ewan Klein and Ivan Sag (1985), Type-driven translation, *Linguistics and Philosophy*, 8(2):163–201.

Jayant Krishnamurthy and Tom Mitchell (2013), 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, pp. 1–10, Association for Computational Linguistics, Stroudsburg, PA, http://aclweb.org/anthology/W13-3201.

Mike LEWIS and Mark STEEDMAN (2013), Combined distributional and logical semantics, *Transactions of the Association for Computational Linguistics*, 1:179–192, http://aclweb.org/anthology/Q13-1015.

Jean MAILLARD, Stephen CLARK, and Edward GREFENSTETTE (2014), A type-driven tensor-based semantics for CCG, in Robin COOPER, Simon DOBNIK, Shalom LAPPIN, and Staffan LARSSON, editors, *Proceedings of the EACL 2014 on Type Theory and Natural Language Semantics (TTNLS)*, pp. 46–54, Association for Computational Linguistics, Stroudsburg, PA, http://aclweb.org/anthology/W14-1406.

Jeff MITCHELL and Mirella LAPATA (2010), Composition in distributional models of semantics, *Cognitive Science*, 34(8):1388–1439.

Richard MONTAGUE (1974), The proper treatment of quantification in ordinary English, in Richmond THOMASON, editor, *Formal Philosophy. Selected Papers of Richard Montague*, pp. 247–270, Yale University Press, New Haven, CT.

Reinhard MUSKENS (2001), Categorial grammar and lexical-functional grammar, in Miriam BUTT and Tracy Holloway KING, editors, *Proceedings of the LFG01 Conference, University of Hong Kong*, pp. 259–279, CSLI, Stanford, CA, http://cslipublications.stanford.edu/LFG/6/lfg01.html.

Reinhard MUSKENS (2003), Language, lambdas, and logic, in Geert-Jan KRUIJFF and Richard OEHRLE, editors, *Resource-Sensitivity, Binding and Anaphora*, Studies in Linguistics and Philosophy, pp. 23–54, Kluwer, Dordrecht.

Reinhard MUSKENS (2010), New directions in type-theoretic grammars, *Journal of Logic, Language and Information*, 19(2):129–136.

Reinhard MUSKENS and Mehrnoosh SADRZADEH (2016a), Context update for lambdas and vectors, in Maxime AMBLARD, Philippe DE GROOTE, Sylvain POGODALLA, and Christian RETORÉ, editors, *Proceedings of the 9th International Conference on Logical Aspects of Computational Linguistics (LACL 2016)*, volume 10054 of *LNCS*, pp. 247–254, Springer-Verlag, Berlin, Heidelberg.

Reinhard MUSKENS and Mehrnoosh SADRZADEH (2016b), Lambdas and vectors, in *Workshop on Distributional Semantics and Linguistic Theory (DSALT), 28th European Summer School in Logic, Language and Information (ESSLLI)*, Free University of Bozen-Bolzano.

Reinhard MUSKENS, Johan VAN BENTHEM, and Albert VISSER (1997), Dynamics, in Johan VAN BENTHEM and Alice TER MEULEN, editors, *Handbook of Logic and Language*, pp. 587–648, Elsevier.

Barbara PARTEE (1986), Noun phrase interpretation and type-shifting principles, in Jeroen GROENENDIJK, Dick DE JONGH, and Martin STOKHOF,

editors, *Studies in Discourse Representation and the Theory of Generalized Quantifiers*, pp. 115–143, Foris, Dordrecht.

Hoifung POON and Pedro DOMINGOS (2009), Unsupervised semantic parsing, in Philipp KOEHN and Rada MIHALCEA, editors, *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-9): Volume 1*, pp. 1–10, Association for Computational Linguistics, Stroudsburg, PA, ISBN 978-1-932432-59-6, http://aclweb.org/anthology/D09-1001.

Paul PORTNER and Barbara PARTEE (2002), *Formal Semantics: The Essential Readings*, Blackwell, Oxford.

Sebastian RIEDEL, Limin YAO, and Andrew MCCALLUM (2010), Modeling relations and their mentions without labeled text, in José Luis BALCÁZAR, Francesco BONCHI, Aristides GIONIS, and Michèle SEBAG, editors, *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD'10): Part III*, volume 6323 of *LNAI*, pp. 148–163, Springer-Verlag, Berlin, Heidelberg, ISBN 3-642-15938-9, 978-3-642-15938-1, http://dl.acm.org/citation.cfm?id=1889788.1889799.

Herbert RUBENSTEIN and John GOODENOUGH (1965), Contextual correlates of synonymy, *Communications of the ACM*, 8(10):627–633.

Mark STEEDMAN (2000), *The Syntactic Process*, MIT Press.

Alfred TARSKI (1965), *Introduction to Logic and to the Methodology of Deductive Sciences*, Oxford University Press, Oxford, 3rd edition.

Frank VELTMAN (1996), Defaults in update semantics, *Journal of Philosophical Logic*, 25(3):221–261.

Limin YAO, Sebastian RIEDEL, and Andrew MCCALLUM (2012), Unsupervised relation discovery with sense disambiguation, in Haizhou LI, Chin-Yew LIN, Miles OSBORNE, Gary Geunbae LEE, and Jong C. PARK, editors, *Proceedings of the 50th annual meeting of the Association for Computational Linguistics: long papers – Volume 1*, ACL '12, pp. 712–720, Association for Computational Linguistics, Stroudsburg, PA, http://aclweb.org/anthology/P12-1075.

Naomi ZEICHNER, Jonathan BERANT, and Ido DAGAN (2012), Crowdsourcing inference-rule evaluation, in Haizhou LI, Chin-Yew LIN, Miles OSBORNE, Gary Geunbae LEE, and Jong C. PARK, editors, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers – Volume 2*, ACL '12, pp. 156–160, Association for Computational Linguistics, Stroudsburg, PA, http://aclweb.org/anthology/P12-2031.