Compositional Matrix-Space Models of Language

Sebastian Rudolph
Karlsruhe Institute of Technology
Karlsruhe, Germany
rudolph@kit.edu

Eugenie Giesbrecht
FZI Forschungszentrum Informatik
Karlsuhe, Germany
giesbrecht@fzi.de

Abstract

We propose CMSMs, a novel type of generic compositional models for syntactic and semantic aspects of natural language, based on matrix multiplication. We argue for the structural and cognitive plausibility of this model and show that it is able to cover and combine various common compositional NLP approaches ranging from statistical word space models to symbolic grammar formalisms.

1 Introduction

In computational linguistics and information retrieval, Vector Space Models (Salton et al., 1975) and its variations – such as Word Space Models (Schütze, 1993), Hyperspace Analogue to Language (Lund and Burgess, 1996), or Latent Semantic Analysis (Deerwester et al., 1990) – have become a mainstream paradigm for text representation. Vector Space Models (VSMs) have been empirically justified by results from cognitive science (Gärdenfors, 2000). They embody the distributional hypothesis of meaning (Firth, 1957), according to which the meaning of words is defined by contexts in which they (co-)occur. Depending on the specific model employed, these contexts can be either local (the co-occurring words), or global (a sentence or a paragraph or the whole document). Indeed, VSMs proved to perform well in a number of tasks requiring computation of *semantic relatedness* between words, such as synonymy identification (Landauer and Dumais, 1997), automatic thesaurus construction (Grefenstette, 1994), semantic priming, and word sense disambiguation (Padó and Lapata, 2007).

Until recently, little attention has been paid to the task of modeling more complex conceptual structures with such models, which constitutes a crucial barrier for semantic vector models on the way to model language (Widdows, 2008). An emerging area of research receiving more and more attention among the advocates of distributional models addresses the methods, algorithms, and evaluation strategies for representing *compositional* aspects of language within a VSM framework. This requires novel modeling paradigms, as most VSMs have been predominantly used for meaning representation of single words and the key problem of common bag-of-words-based VSMs is that word order information and thereby the structure of the language is lost.

There are approaches under way to work out a combined framework for meaning representation using both the advantages of symbolic and distributional methods. Clark and Pulman (2007) suggest a conceptual model which unites symbolic and distributional representations by means of traversing the parse tree of a sentence and applying a tensor product for combining vectors of the meanings of words with the vectors of their roles. The model is further elaborated by Clark et al. (2008).

To overcome the aforementioned difficulties with VSMs and work towards a tight integration of symbolic and distributional approaches, we propose a *Compositional Matrix-Space Model* (CMSM) which employs matrices instead of vectors and makes use of matrix multiplication as the one and only composition operation.

The paper is structured as follows: We start by providing the necessary basic notions in linear algebra in Section 2. In Section 3, we give a formal account of the concept of compositionality, introduce our model, and argue for the plausibility of CMSMs in the light of structural and cognitive considerations. Section 4 shows how common VSM approaches to compositionality can be captured by CMSMs while Section 5 illustrates the capabilities of our model to likewise cover symbolic approaches. In Section 6, we demonstrate

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 907–916,
Uppsala, Sweden, 11-16 July 2010.

how several CMSMs can be combined into one model. We provide an overview of related work in Section 7 before we conclude and point out avenues for further research in Section 8.

2 Preliminaries

In this section, we recap some aspects of linear algebra to the extent needed for our considerations about CMSMs. For a more thorough treatise we refer the reader to a linear algebra textbook (such as Strang (1993)).

Vectors. Given a natural number n, an n-dimensional vector $\mathbf{v}$ over the reals can be seen as a list (or tuple) containing n real numbers $r_1, \ldots, r_n \in \mathbb{R}$, written $\mathbf{v} = (r_1 \quad r_2 \quad \cdots \quad r_n)$. Vectors will be denoted by lowercase bold font letters and we will use the notation $\mathbf{v}(i)$ to refer to the ith entry of vector $\mathbf{v}$. As usual, we write $\mathbb{R}^n$ to denote the set of all n-dimensional vectors with real entries. Vectors can be added entry-wise, i.e., $(r_1 \quad \cdots \quad r_n) + (r'_1 \quad \cdots \quad r'_n) = (r_1 + r'_1 \quad \cdots \quad r_n + r'_n)$. Likewise, the entry-wise product (also known as Hadamard product) is defined by $(r_1 \quad \cdots \quad r_n) \odot (r'_1 \quad \cdots \quad r'_n) = (r_1 \cdot r'_1 \quad \cdots \quad r_n \cdot r'_n)$.

Matrices. Given two real numbers n, m, an $n \times m$ matrix over the reals is an array of real numbers with n rows and m columns. We will use capital letters to denote matrices and, given a matrix M we will write $M(i, j)$ to refer to the entry in the ith row and the jth column:

$$M = \begin{pmatrix} M(1,1) & M(1,2) & \cdots & M(1,j) & \cdots & M(1,m) \\ M(2,1) & M(2,2) & & & & \vdots \\ \vdots & & & & & \vdots \\ M(i,1) & & & M(i,j) & & \vdots \\ \vdots & & & & & \vdots \\ M(n,1) & M(1,2) & \cdots & \cdots & \cdots & M(n,m) \end{pmatrix}$$

The set of all $n \times m$ matrices with real number entries is denoted by $\mathbb{R}^{n \times m}$. Obviously, m-dimensional vectors can be seen as $1 \times m$ matrices. A matrix can be *transposed* by exchanging columns and rows: given the $n \times m$ matrix M, its transposed version M^T is a $m \times n$ matrix defined by $M^T(i, j) = M(j, i)$.

Linear Mappings. Beyond being merely array-like data structures, matrices correspond to certain type of functions, so-called *linear mappings*, having vectors as in- and output. More precisely, an $n \times m$ matrix M applied to an m-dimensional vector $\mathbf{v}$ yields an n-dimensional vector $\mathbf{v}'$ (written: $\mathbf{v}M = \mathbf{v}'$) according to

$$\mathbf{v}'(i) = \sum_{j=1}^{m} \mathbf{v}(j) \cdot M(i, j)$$

Linear mappings can be concatenated, giving rise to the notion of standard matrix multiplication: we write $M_1 M_2$ to denote the matrix that corresponds to the linear mapping defined by applying first M_1 and then M_2. Formally, the matrix product of the $n \times l$ matrix M_1 and the $l \times m$ matrix M_2 is an $n \times m$ matrix $M = M_1 M_2$ defined by

$$M(i, j) = \sum_{k=1}^{l} M_1(i, k) \cdot M_2(k, j)$$

Note that the matrix product is associative (i.e., $(M_1 M_2)M_3 = M_1(M_2 M_3)$ always holds, thus parentheses can be omitted) but not commutative ($M_1 M_2 = M_2 M_1$ does not hold in general, i.e., the order matters).

Permutations. Given a natural number n, a *permutation* on $\{1 \ldots n\}$ is a bijection (i.e., a mapping that is one-to-one and onto) $\Phi : \{1 \ldots n\} \to \{1 \ldots n\}$. A permutation can be seen as a "reordering scheme" on a list with n elements: the element at position i will get the new position $\Phi(i)$ in the reordered list. Likewise, a permutation can be applied to a vector resulting in a rearrangement of the entries. We write Φ^n to denote the permutation corresponding to the n-fold application of Φ and Φ^{-1} to denote the permutation that "undoes" Φ.

Given a permutation Φ, the corresponding *permutation matrix* M_Φ is defined by

$$M_\Phi(i, j) = \begin{cases} 1 \text{ if } \Phi(j) = i, \\ 0 \text{ otherwise.} \end{cases}$$

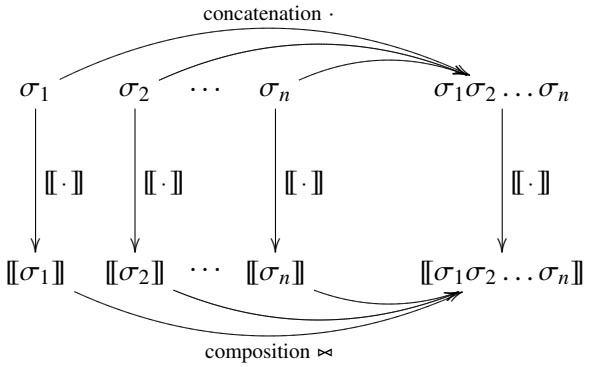
Then, obviously permuting a vector according to Φ can be expressed in terms of matrix multiplication as well as we obtain for any vector $\mathbf{v} \in \mathbb{R}^n$:

$$\Phi(\mathbf{v}) = \mathbf{v} M_\Phi$$

Likewise, iterated application (Φ^n) and the inverses Φ^{-n} carry over naturally to the corresponding notions in matrices.

3 Compositionality and Matrices

The underlying principle of compositional semantics is that the meaning of a sentence (or a word phrase) can be derived from the meaning of its constituent tokens by applying a composition operation. More formally, the underlying idea can be described as follows: given a mapping $[\![\cdot]\!] : \Sigma \to \mathbb{S}$ from a set of tokens (words) Σ into some semantical space $\mathbb{S}$ (the elements of which we will simply call "meanings"), we find a semantic composition operation $\bowtie$: $\mathbb{S}^* \to \mathbb{S}$ mapping sequences of meanings to meanings such that the meaning of a sequence of tokens $\sigma_1\sigma_2 \ldots \sigma_n$ can be obtained by applying $\bowtie$ to the sequence $[\![\sigma_1]\!][\![\sigma_2]\!] \ldots [\![\sigma_n]\!]$. This situation qualifies $[\![\cdot]\!]$ as a homomorphism between $(\Sigma^*, \cdot)$ and $(\mathbb{S}, \bowtie)$ and can be displayed as follows:



A great variety of linguistic models are subsumed by this general idea ranging from purely symbolic approaches (like type systems and categorial grammars) to rather statistical models (like vector space and word space models). At the first glance, the underlying encodings of word semantics as well as the composition operations differ significantly. However, we argue that a great variety of them can be incorporated – and even freely inter-combined – into a unified model where the semantics of simple tokens and complex phrases is expressed by matrices and the composition operation is standard matrix multiplication.

More precisely, in Compositional Marix-Space Models, we have $\mathbb{S} = \mathbb{R}^{n \times n}$, i.e. the semantical space consists of quadratic matrices, and the composition operator $\bowtie$ coincides with matrix multiplication as introduced in Section 2. In the following, we will provide diverse arguments illustrating that CMSMs are intuitive and natural.

3.1 Algebraic Plausibility – Structural Operation Properties

Most linear-algebra-based operations that have been proposed to model composition in language models are associative and commutative. Thereby, they realize a multiset (or bag-of-words) semantics that makes them insensitive to structural differences of phrases conveyed through word order.

While associativity seems somewhat acceptable and could be defended by pointing to the stream-like, sequential nature of language, commutativity seems way less justifiable, arguably.

As mentioned before, matrix multiplication is associative but non-commutative, whence we propose it as more adequate for modeling compositional semantics of language.

3.2 Neurological Plausibility – Progression of Mental States

From a very abstract and simplified perspective, CMSMs can also be justified neurologically.

Suppose the mental state of a person at one specific moment in time can be encoded by a vector $\mathbf{v}$ of numerical values; one might, e.g., think of the level of excitation of neurons. Then, an external stimulus or signal, such as a perceived word, will result in a change of the mental state. Thus, the external stimulus can be seen as a function being applied to $\mathbf{v}$ yielding as result the vector $\mathbf{v}'$ that corresponds to the persons mental state after receiving the signal. Therefore, it seems sensible to associate with every signal (in our case: token σ) a respective function (a linear mapping, represented by a matrix $M = [\![\sigma]\!]$ that maps mental states to mental states (i.e. vectors $\mathbf{v}$ to vectors $\mathbf{v}' = \mathbf{v}M$).

Consequently, the subsequent reception of inputs σ, σ' associated to matrices M and M' will transform a mental vector $\mathbf{v}$ into the vector $(\mathbf{v}M)M'$ which by associativity equals $\mathbf{v}(MM')$. Therefore, MM' represents the mental state transition triggered by the signal sequence $\sigma\sigma'$. Naturally, this consideration carries over to sequences of arbitrary length. This way, abstracting from specific initial mental state vectors, our semantic space $\mathbb{S}$ can be seen as a function space of mental transformations represented by matrices, whereby matrix multiplication realizes subsequent execution of those transformations triggered by the input token sequence.

3.3 Psychological Plausibility – Operations on Working Memory

A structurally very similar argument can be provided on another cognitive explanatory level. There have been extensive studies about human language processing justifying the hypothesis of a *working memory* (Baddeley, 2003). The mental state vector can be seen as representation of a person's working memory which gets transformed by external input. Note that matrices can perform standard memory operations such as storing, deleting, copying etc. For instance, the matrix $M_{\text{copy(k,l)}}$ defined by

$$M_{\text{copy(k,l)}}(i,j) = \begin{cases} 1 \text{ if } i = j \neq l \text{ or } i = k,\ j = l, \\ 0 \text{ otherwise.} \end{cases}$$

applied to a vector $\mathbf{v}$, will copy its kth entry to the lth position. This mechanism of storage and insertion can, e.g., be used to simulate simple forms of anaphora resolution.

4 CMSMs Encode Vector Space Models

In VSMs numerous vector operations have been used to model composition (Widdows, 2008), some of the more advanced ones being related to quantum mechanics. We show how these common composition operators can be modeled by CMSMs.[1] Given a vector composition operation $\bowtie\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, we provide a surjective function $\psi_\bowtie : \mathbb{R}^n \to \mathbb{R}^{n' \times n'}$ that translates the vector representation into a matrix representation in a way such that for all $\mathbf{v}_1, \ldots \mathbf{v}_k \in \mathbb{R}^n$ holds

$$\mathbf{v}_1 \bowtie \ldots \bowtie \mathbf{v}_k = \psi_\bowtie^{-1}(\psi_\bowtie(\mathbf{v}_1) \ldots \psi_\bowtie(\mathbf{v}_k))$$

where $\psi_\bowtie(\mathbf{v}_i)\psi_\bowtie(\mathbf{v}_j)$ denotes matrix multiplication of the matrices assigned to $\mathbf{v}_i$ and $\mathbf{v}_j$.

4.1 Vector Addition

As a simple basic model for semantic composition, vector addition has been proposed. Thereby, tokens σ get assigned (usually high-dimensional) vectors $\mathbf{v}_\sigma$ and to obtain a representation of the meaning of a phrase or a sentence $w = \sigma_1 \ldots \sigma_k$, the vector sum of the vectors associated to the constituent tokens is calculated: $\mathbf{v}_w = \sum_{i=1}^k \mathbf{v}_{\sigma_i}$.

[1] In our investigations we will focus on VSM composition operations which preserve the format (i.e. which yield a vector of the same dimensionality), as our notion of compositionality requires models that allow for iterated composition. In particular, this rules out dot product and tensor product. However the convolution product can be seen as a condensed version of the tensor product.

This kind of composition operation is subsumed by CMSMs; suppose in the original model, a token σ gets assigned the vector $\mathbf{v}_\sigma$, then by defining

$$\psi_+(\mathbf{v}_\sigma) = \left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline & \mathbf{v}_\sigma & & 1 \end{array}\right)$$

(mapping n-dimensional vectors to $(n+1)\times(n+1)$ matrices), we obtain for a phrase $w = \sigma_1 \ldots \sigma_k$

$$\psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}) \ldots \psi_+(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{\sigma_1} + \ldots + \mathbf{v}_{\sigma_k} = \mathbf{v}_w.$$

Proof. By induction on k. For $k = 1$, we have $\mathbf{v}_w = \mathbf{v}_\sigma = \psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}))$. For $k > 1$, we have

$$\begin{aligned} & \psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}) \ldots \psi_+(\mathbf{v}_{\sigma_{k-1}})\psi_+(\mathbf{v}_{\sigma_k})) \\ = \ & \psi_+^{-1}(\psi_+(\psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}) \ldots \psi_+(\mathbf{v}_{\sigma_{k-1}})))\psi_+(\mathbf{v}_{\sigma_k})) \\ \overset{i.h.}{=} \ & \psi_+^{-1}(\psi_+(\textstyle\sum_{i=1}^{k-1} \mathbf{v}_{\sigma_i})\psi_+(\mathbf{v}_{\sigma_k})) \end{aligned}$$

$$= \psi_+^{-1}\left(\left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline \sum_{i=1}^{k-1}\mathbf{v}_{\sigma_i}(1) & \cdots & \sum_{i=1}^{k-1}\mathbf{v}_{\sigma_i}(n) & 1 \end{array}\right)\left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline \mathbf{v}_{\sigma_k}(1) & \cdots & \mathbf{v}_{\sigma_k}(n) & 1 \end{array}\right)\right)$$

$$= \psi_+^{-1}\left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline \sum_{i=1}^{k}\mathbf{v}_{\sigma_i}(1) & \cdots & \sum_{i=1}^{k}\mathbf{v}_{\sigma_i}(n) & 1 \end{array}\right) = \sum_{i=1}^{k}\mathbf{v}_{\sigma_i} \qquad q.e.d.[2]$$

4.2 Component-wise Multiplication

On the other hand, the Hadamard product (also called entry-wise product, denoted by $\odot$) has been proposed as an alternative way of semantically composing token vectors.

By using a different encoding into matrices, CMSMs can simulate this type of composition operation as well. By letting

$$\psi_\odot(\mathbf{v}_\sigma) = \begin{pmatrix} \mathbf{v}_\sigma(1) & 0 & \cdots & 0 \\ 0 & \mathbf{v}_\sigma(2) & & \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{v}_\sigma(n) \end{pmatrix},$$

we obtain an $n \times n$ matrix representation for which $\psi_\odot^{-1}(\psi_\odot(\mathbf{v}_{\sigma_1}) \ldots \psi_\odot(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{\sigma_1} \odot \ldots \odot \mathbf{v}_{\sigma_k} = \mathbf{v}_w$.

4.3 Holographic Reduced Representations

Holographic reduced representations as introduced by Plate (1995) can be seen as a refinement

[2] The proofs for the respective correspondences for $\odot$ and $\circledast$ as well as the permutation-based approach in the following sections are structurally analog, hence, we will omit them for space reasons.

of convolution products with the benefit of preserving dimensionality: given two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^n$, their *circular convolution product* $\mathbf{v}_1 \circledast \mathbf{v}_2$ is again an n-dimensional vector $\mathbf{v}_3$ defined by

$$\mathbf{v}_3(i+1) = \sum_{k=0}^{n-1} \mathbf{v}_1(k+1) \cdot \mathbf{v}_2((i-k \mod n)+1)$$

for $0 \leq i \leq n-1$. Now let $\psi_{\circledast}(\mathbf{v})$ be the $n \times n$ matrix M with

$$M(i,j) = \mathbf{v}((j-i \mod n)+1).$$

In the 3-dimensional case, this would result in

$$\psi_{\circledast}(\mathbf{v}(1) \quad \mathbf{v}(2) \quad \mathbf{v}(3)) = \begin{pmatrix} \mathbf{v}(1) & \mathbf{v}(2) & \mathbf{v}(3) \\ \mathbf{v}(3) & \mathbf{v}(1) & \mathbf{v}(2) \\ \mathbf{v}(2) & \mathbf{v}(3) & \mathbf{v}(1) \end{pmatrix}$$

Then, it can be readily checked that

$$\psi_{\circledast}^{-1}(\psi_{\circledast}(\mathbf{v}_{\sigma_1}) \ldots \psi_{\circledast}(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{\sigma_1} \circledast \ldots \circledast \mathbf{v}_{\sigma_k} = \mathbf{v}_w.$$

4.4 Permutation-based Approaches

Sahlgren et al. (2008) use permutations on vectors to account for word order. In this approach, given a token σ_m occurring in a sentence $w = \sigma_1 \ldots \sigma_k$ with predefined "uncontextualized" vectors $\mathbf{v}_{\sigma_1} \ldots \mathbf{v}_{\sigma_k}$, we compute the contextualized vector $\mathbf{v}_{w,m}$ for σ_m by

$$\mathbf{v}_{w,m} = \Phi^{1-m}(\mathbf{v}_{\sigma_1}) + \ldots + \Phi^{k-m}(\mathbf{v}_{\sigma_k}),$$

which can be equivalently transformed into

$$\Phi^{1-m}(\mathbf{v}_{\sigma_1} + \Phi(\ldots + \Phi(\mathbf{v}_{\sigma_{k-1}} + (\Phi(\mathbf{v}_{\sigma_k}))) \ldots)).$$

Note that the approach is still token-centered, i.e., a vector representation of a token is endowed with contextual representations of surrounding tokens. Nevertheless, this setting can be transferred to a CMSM setting by recording the position of the focused token as an additional parameter. Now, by assigning every $\mathbf{v}_\sigma$ the matrix

$$\psi_\Phi(\mathbf{v}_\sigma) = \left(\begin{array}{c|c} M_\Phi & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline \mathbf{v}_\sigma & 1 \end{array}\right)$$

we observe that for

$$M_{w,m} := (M_\Phi^-)^{m-1} \psi_\Phi(\mathbf{v}_{\sigma_1}) \ldots \psi_\Phi(\mathbf{v}_{\sigma_k})$$

we have

$$M_{w,m} = \left(\begin{array}{c|c} M_\Phi^{k-m} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline \mathbf{v}_{w,m} & 1 \end{array}\right),$$

whence $\psi_\Phi^{-1}((M_\Phi^-)^{m-1} \psi_\Phi(\mathbf{v}_{\sigma_1}) \ldots \psi_\Phi(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{w,m}$.

5 CMSMs Encode Symbolic Approaches

Now we will elaborate on symbolic approaches to language, i.e., discrete grammar formalisms, and show how they can conveniently be embedded into CMSMs. This might come as a surprise, as the apparent likeness of CMSMs to vector-space models may suggest incompatibility to discrete settings.

5.1 Group Theory

Group theory and grammar formalisms based on groups and pre-groups play an important role in computational linguistics (Dymetman, 1998; Lambek, 1958). From the perspective of our compositionality framework, those approaches employ a group (or pre-group) $(G, \cdot)$ as semantical space $\mathbb{S}$ where the group operation (often written as multiplication) is used as composition operation $\bowtie$.

According Cayley's Theorem (Cayley, 1854), every group G is isomorphic to a permutation group on some set S. Hence, assuming finiteness of G and consequently S, we can encode group-based grammar formalisms into CMSMs in a straightforward way by using permutation matrices of size $|S| \times |S|$.

5.2 Regular Languages

Regular languages constitute a basic type of languages characterized by a symbolic formalism. We will show how to select the assignment $[\![\cdot]\!]$ for a CMSM such that the matrix associated to a token sequence exhibits whether this sequence belongs to a given regular language, that is if it is accepted by a given finite state automaton. As usual (cf. e.g., Hopcroft and Ullman (1979)) we define a nondeterministic finite automaton $\mathcal{A} = (Q, \Sigma, \Delta, Q_\mathrm{I}, Q_\mathrm{F})$ with $Q = \{q_0, \ldots, q_{n-1}\}$ being the set of states, Σ the input alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ the transition relation, and Q_I and Q_F being the sets of initial and final states, respectively.

Then we assign to every token $\sigma \in \Sigma$ the $n \times n$ matrix $[\![\sigma]\!] = M$ with

$$M(i,j) = \begin{cases} 1 \text{ if } (q_i, \sigma, q_j) \in \Delta, \\ 0 \text{ otherwise.} \end{cases}$$

Hence essentially, the matrix M encodes all state transitions which can be caused by the input σ. Likewise, for a word $w = \sigma_1 \dots \sigma_k \in \Sigma^*$, the matrix $M_w := [\![\sigma_1]\!] \dots [\![\sigma_k]\!]$ will encode all state transitions mediated by w. Finally, if we define vectors $\mathbf{v}_\mathrm{I}$ and $\mathbf{v}_\mathrm{F}$ by

$$\mathbf{v}_\mathrm{I}(i) = \begin{cases} 1 \text{ if } q_i \in Q_\mathrm{I}, \\ 0 \text{ otherwise,} \end{cases} \qquad \mathbf{v}_\mathrm{F}(i) = \begin{cases} 1 \text{ if } q_i \in Q_\mathrm{F}, \\ 0 \text{ otherwise,} \end{cases}$$

then we find that w is accepted by $\mathcal{A}$ exactly if $\mathbf{v}_\mathrm{I} M_w \mathbf{v}_\mathrm{F}^T \geq 1$.

5.3 The General Case: Matrix Grammars

Motivated by the above findings, we now define a general notion of matrix grammars as follows:

Definition 1 *Let Σ be an alphabet. A* matrix grammar *$\mathcal{M}$ of degree n is defined as the pair $\langle [\![\cdot]\!], AC\rangle$ where $[\![\cdot]\!]$ is a mapping from Σ to $n \times n$ matrices and $AC = \{\langle \mathbf{v}'_1, \mathbf{v}_1, r_1\rangle, \dots, \langle \mathbf{v}'_m, \mathbf{v}_m, r_m\rangle\}$ with $\mathbf{v}'_1, \mathbf{v}_1, \dots, \mathbf{v}'_m, \mathbf{v}_m \in \mathbb{R}^n$ and $r_1, \dots, r_m \in \mathbb{R}$ is a finite set of* acceptance conditions. *The language generated by $\mathcal{M}$ (denoted by $L(\mathcal{M})$) contains a token sequence $\sigma_1 \dots \sigma_k \in \Sigma^*$ exactly if $\mathbf{v}'_i [\![\sigma_1]\!] \dots [\![\sigma_k]\!] \mathbf{v}_i^T \geq r_i$ for all $i \in \{1, \dots, m\}$. We will call a language L* matricible *if $L = L(\mathcal{M})$ for some matrix grammar $\mathcal{M}$.*

Then, the following proposition is a direct consequence from the preceding section.

Proposition 1 *Regular languages are matricible.*

However, as demonstrated by the subsequent examples, also many non-regular and even non-context-free languages are matricible, hinting at the expressivity of our grammar model.

Example 1 *We define $\mathcal{M}\langle [\![\cdot]\!], AC\rangle$ with*

$$\Sigma = \{a, b, c\} \qquad [\![a]\!] = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$[\![b]\!] = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad [\![c]\!] = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 3 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{aligned} AC = \{ &\langle (0 \quad 0 \quad 1 \quad 1), (1 \quad -1 \quad 0 \quad 0), 0\rangle, \\ &\langle (0 \quad 0 \quad 1 \quad 1), (-1 \quad 1 \quad 0 \quad 0), 0\rangle\} \end{aligned}$$

Then $L(\mathcal{M})$ contains exactly all palindromes from $\{a, b, c\}^*$, i.e., the words $d_1 d_2 \dots d_{n-1} d_n$ for which $d_1 d_2 \dots d_{n-1} d_n = d_n d_{n-1} \dots d_2 d_1$.

Example 2 *We define $\mathcal{M} = \langle [\![\cdot]\!], AC\rangle$ with*

$$\Sigma = \{a, b, c\} \qquad [\![a]\!] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$[\![b]\!] = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad [\![c]\!] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

$$\begin{aligned} AC = \{ &\langle (1 \ 0 \ 0 \ 0 \ 0 \ 0), (0 \ 0 \ 1 \ 0 \ 0 \ 0), 1\rangle, \\ &\langle (0 \ 0 \ 0 \ 1 \ 1 \ 0), (0 \ 0 \ 0 \ 1 \ -1 \ 0), 0\rangle, \\ &\langle (0 \ 0 \ 0 \ 0 \ 1 \ 1), (0 \ 0 \ 0 \ 0 \ 1 \ -1), 0\rangle, \\ &\langle (0 \ 0 \ 0 \ 1 \ 1 \ 0), (0 \ 0 \ 0 \ -1 \ 0 \ 1), 0\rangle\} \end{aligned}$$

Then $L(\mathcal{M})$ is the (non-context-free) language $\{a^m b^m c^m \mid m > 0\}$.

The following properties of matrix grammars and matricible language are straightforward.

Proposition 2 *All languages characterized by a set of linear equations on the letter counts are matricible.*

Proof. Suppose $\Sigma = \{a_1, \dots a_n\}$. Given a word w, let x_i denote the number of occurrences of a_i in w. A linear equation on the letter counts has the form

$$k_1 x_1 + \dots + k_n x_n = k \qquad (k, k_1, \dots, k_n \in \mathbb{R})$$

Now define $[\![a_i]\!] = \psi_+(\mathbf{e}_i)$, where $\mathbf{e}_i$ is the ith unit vector, i.e. it contains a 1 at he ith position and 0 in all other positions. Then, it is easy to see that w will be mapped to $M = \psi_+(x_1 \quad \cdots \quad x_n)$. Due to the fact that $\mathbf{e}_{n+1} M = (x_1 \quad \cdots \quad x_n \quad 1)$ we can enforce the above linear equation by defining the acceptance conditions

$$\begin{aligned} AC = \{ &\langle \mathbf{e}_{n+1}, (k_1 \quad \dots \quad k_n \quad -k), 0\rangle, \\ &\langle -\mathbf{e}_{n+1}, (k_1 \quad \dots \quad k_n \quad -k), 0\rangle\}. \end{aligned}$$

q.e.d.

Proposition 3 *The intersection of two matricible languages is again a matricible language.*

Proof. This is a direct consequence of the considerations in Section 6 together with the observation, that the new set of acceptance conditions is trivially obtained from the old ones with adapted dimensionalities. *q.e.d.*

Note that the fact that the language $\{a^m b^m c^m \mid m > 0\}$ is matricible, as demonstrated in Example 2 is a straightforward consequence of the Propositions 1, 2, and 3, since the language in question can be described as the intersection of the regular language $a^+b^+c^+$ with the language characterized by the equations $x_a - x_b = 0$ and $x_b - x_c = 0$. We proceed by giving another account of the expressivity of matrix grammars by showing undecidability of the emptiness problem.

Proposition 4 *The problem whether there is a word which is accepted by a given matrix grammar is undecidable.*

Proof. The undecidable *Post correspondence problem* (Post, 1946) is described as follows: given two lists of words $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ over some alphabet Σ', is there a sequence of numbers $h_1, \ldots, h_m$ $(1 \leq h_j \leq n)$ such that $u_{h_1} \ldots u_{h_m} = v_{h_1} \ldots v_{h_m}$?

We now reduce this problem to the emptiness problem of a matrix grammar. W.l.o.g., let $\Sigma' = \{a_1, \ldots, a_k\}$. We define a bijection # from Σ'^* to $\mathbb{N}$ by

$$\#(a_{n_1} a_{n_2} \ldots a_{n_l}) = \sum_{i=1}^{l} (n_i - 1) \cdot k^{(l-i)}$$

Note that this is indeed a bijection and that for $w_1, w_2 \in \Sigma'^*$, we have

$$\#(w_1 w_2) = \#(w_1) \cdot k^{|w_2|} + \#(w_2).$$

Now, we define $\mathcal{M}$ as follows:

$$\Sigma = \{b_1, \ldots b_n\} \qquad [\![b_i]\!] = \begin{pmatrix} k^{|u_i|} & 0 & 0 \\ 0 & k^{|v_i|} & 0 \\ \#(u_i) & \#(v_i) & 1 \end{pmatrix}$$

$$AC = \{ \langle (0 \quad 0 \quad 1), (1 \quad -1 \quad 0), 0 \rangle, \\ \langle (0 \quad 0 \quad 1), (-1 \quad 1 \quad 0), 0 \rangle \}$$

Using the above fact about # and a simple induction on m, we find that

$$[\![a_{h_1}]\!] \ldots [\![a_{h_m}]\!] = \begin{pmatrix} k^{|u_{h_1} .. u_{h_m}|} & 0 & 0 \\ 0 & k^{|v_{h_1} .. v_{h_m}|} & 0 \\ \#(u_{h_1} \ldots u_{h_m}) & \#(v_{h_1} \ldots v_{h_m}) & 1 \end{pmatrix}$$

Evaluating the two acceptance conditions, we find them satisfied exactly if $\#(u_{h_1} \ldots u_{h_m}) = \#(v_{h_1} \ldots v_{h_m})$. Since # is a bijection, this is the case if and only if $u_{h_1} \ldots u_{h_m} = v_{h_1} \ldots v_{h_m}$. Therefore $\mathcal{M}$ accepts $b_{h_1} \ldots b_{h_m}$ exactly if the sequence $h_1, \ldots, h_m$ is a solution to the given Post Correspondence Problem. Consequently, the question whether such a solution exists is equivalent to the question whether the language $L(\mathcal{M})$ is non-empty. *q.e.d.*

These results demonstrate that matrix grammars cover a wide range of formal languages. Nevertheless some important questions remain open and need to be clarified next:

Are all context-free languages matricible? We conjecture that this is not the case.[3] Note that this question is directly related to the question whether Lambek calculus can be modeled by matrix grammars.

Are matricible languages closed under concatenation? That is: given two arbitrary matricible languages L_1, L_2, is the language $L = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$ again matricible? Being a property common to all language types from the Chomsky hierarchy, answering this question is surprisingly non-trivial for matrix grammars.

In case of a negative answer to one of the above questions it might be worthwhile to introduce an extended notion of context grammars to accommodate those desirable properties. For example, allowing for some nondeterminism by associating several matrices to one token would ensure closure under concatenation.

How do the theoretical properties of matrix grammars depend on the underlying algebraic structure? Remember that we considered matrices containing real numbers as entries. In general, matrices can be defined on top of any mathematical structure that is (at least) a semiring (Golan, 1992). Examples for semirings are the natural numbers, boolean algebras, or polynomials with natural number coefficients. Therefore, it would be interesting to investigate the influence of the choice of the underlying semiring on the properties of the matrix grammars – possibly non-standard structures turn out to be more appropriate for capturing certain compositional language properties.

6 Combination of Different Approaches

Another central advantage of the proposed matrix-based models for word meaning is that several matrix models can be easily combined into one.

[3] For instance, we have not been able to find a matrix grammar that recognizes the language of all well-formed parenthesis expressions.

Again assume a sequence $w = \sigma_1 \dots \sigma_k$ of tokens with associated matrices $[\![\sigma_1]\!], \dots, [\![\sigma_k]\!]$ according to one specific model and matrices $(\!|\sigma_1|\!), \dots, (\!|\sigma_k|\!)$ according to another.

Then we can combine the two models into one $\{\!|\ \cdot\ |\!\}$ by assigning to σ_i the matrix

$$\{\!|\sigma_i|\!\} = \left(\begin{array}{ccc|ccc} & & & 0 & \cdots & 0 \\ & [\![\sigma_i]\!] & & \vdots & \ddots & \\ & & & 0 & & 0 \\ \hline 0 & \cdots & 0 & & & \\ \vdots & \ddots & & & (\!|\sigma_i|\!) & \\ 0 & & 0 & & & \end{array}\right)$$

By doing so, we obtain the correspondence

$$\{\!|\sigma_1|\!\} \dots \{\!|\sigma_k|\!\} = \left(\begin{array}{ccc|ccc} & & & 0 & \cdots & 0 \\ & [\![\sigma_1]\!] \dots [\![\sigma_k]\!] & & \vdots & \ddots & \\ & & & 0 & & 0 \\ \hline 0 & \cdots & 0 & & & \\ \vdots & \ddots & & & (\!|\sigma_1|\!) \dots (\!|\sigma_k|\!) & \\ 0 & & 0 & & & \end{array}\right)$$

In other words, the semantic compositions belonging to two CMSMs can be executed "in parallel." Mark that by providing non-zero entries for the upper right and lower left matrix part, information exchange between the two models can be easily realized.

7 Related Work

We are not the first to suggest an extension of classical VSMs to matrices. Distributional models based on matrices or even higher-dimensional arrays have been proposed in information retrieval (Gao et al., 2004; Antonellis and Gallopoulos, 2006). However, to the best of our knowledge, the approach of realizing compositionality via matrix multiplication seems to be entirely original.

Among the early attempts to provide more compelling combinatory functions to capture word order information and the non-commutativity of linguistic compositional operation in VSMs is the work of Kintsch (2001) who is using a more sophisticated addition function to model predicate-argument structures in VSMs.

Mitchell and Lapata (2008) formulate semantic composition as a function $m = f(w_1, w_2, R, K)$ where R is a relation between w_1 and w_2 and K is additional knowledge. They evaluate the model with a number of addition and multiplication operations for vector combination on a sentence similarity task proposed by Kintsch (2001). Widdows (2008) proposes a number of more advanced vector operations well-known from quantum mechanics, such as tensor product and convolution, to model composition in vector spaces. He shows the ability of VSMs to reflect the relational and phrasal meanings on a simplified analogy task. Giesbrecht (2009) evaluates four vector composition operations (+, $\odot$, tensor product, convolution) on the task of identifying multi-word units. The evaluation results of the three studies are not conclusive in terms of which vector operation performs best; the different outcomes might be attributed to the underlying word space models; e.g., the models of Widdows (2008) and Giesbrecht (2009) feature dimensionality reduction while that of Mitchell and Lapata (2008) does not. In the light of these findings, our CMSMs provide the benefit of just one composition operation that is able to mimic all the others as well as combinations thereof.

8 Conclusion and Future Work

We have introduced a generic model for compositionality in language where matrices are associated with tokens and the matrix representation of a token sequence is obtained by iterated matrix multiplication. We have given algebraic, neurological, and psychological plausibility indications in favor of this choice. We have shown that the proposed model is expressive enough to cover and combine a variety of distributional and symbolic aspects of natural language. This nourishes the hope that matrix models can serve as a kind of *lingua franca* for compositional models.

This having said, some crucial questions remain before CMSMs can be applied in practice:
How to acquire CMSMs for large token sets and specific purposes? We have shown the value and expressivity of CMSMs by providing carefully hand-crafted encodings. In practical cases, however, the number of token-to-matrix assignments will be too large for this manual approach. Therefore, methods to (semi-)automatically acquire those assignments from available data are required. To this end, machine learning techniques need to be investigated with respect to their applicability to this task. Presumably, hybrid approaches have to be considered, where parts of

the matrix representation are learned whereas others are stipulated in advance guided by external sources (such as lexical information).

In this setting, data sparsity may be overcome through tensor methods: given a set T of tokens together with the matrix assignment $[\![]\!] : T \rightarrow \mathbb{R}^{n \times n}$, this datastructure can be conceived as a 3-dimensional array (also known as tensor) of size $n \times n \times |T|$ wherein the single token-matrices can be found as slices. Then tensor decomposition techniques can be applied in order to find a compact representation, reduce noise, and cluster together similar tokens (Tucker, 1966; Rendle et al., 2009). First evaluation results employing this approach to the task of free associations are reported by Giesbrecht (2010).

How does linearity limit the applicability of CMSMs? In Section 3, we justified our model by taking the perspective of tokens being functions which realize mental state transitions. Yet, using matrices to represent those functions restricts them to linear mappings. Although this restriction brings about benefits in terms of computability and theoretical accessibility, the limitations introduced by this assumption need to be investigated. Clearly, certain linguistic effects (like a-posteriori disambiguation) cannot be modeled via linear mappings. Instead, we might need some in-between application of simple nonlinear functions in the spirit of quantum-collapsing of a "superposed" mental state (such as the winner takes it all, survival of the top-k vector entries, and so forth). Thus, another avenue of further research is to generalize from the linear approach.

Acknowledgements

This work was supported by the German Research Foundation (DFG) under the Multipla project (grant 38457858) as well as by the German Federal Ministry of Economics (BMWi) under the project Theseus (number 01MQ07019).

References

[Antonellis and Gallopoulos2006] Ioannis Antonellis and Efstratios Gallopoulos. 2006. Exploring term-document matrices from matrix models in text mining. *CoRR*, abs/cs/0602076.

[Baddeley2003] Alan D. Baddeley. 2003. Working memory and language: An overview. *Journal of Communication Disorder*, 36:198–208.

[Cayley1854] Arthur Cayley. 1854. On the theory of groups as depending on the symbolic equation $\theta^n = 1$. *Philos. Magazine*, 7:40–47.

[Clark and Pulman2007] Stephen Clark and Stephen Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction, Stanford, CA, 2007*, pages 52–55.

[Clark et al.2008] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)*, pages 133–140.

[Deerwester et al.1990] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

[Dymetman1998] Marc Dymetman. 1998. Group theory and computational linguistics. *J. of Logic, Lang. and Inf.*, 7(4):461–497.

[Firth1957] John R. Firth. 1957. A synopsis of linguistic theory 1930-55. *Studies in linguistic analysis*, pages 1–32.

[Gao et al.2004] Kai Gao, Yongcheng Wang, and Zhiqi Wang. 2004. An efficient relevant evaluation model in information retrieval and its application. In *CIT '04: Proceedings of the The Fourth International Conference on Computer and Information Technology*, pages 845–850. IEEE Computer Society.

[Gärdenfors2000] Peter Gärdenfors. 2000. *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, MA, USA.

[Giesbrecht2009] Eugenie Giesbrecht. 2009. In search of semantic compositionality in vector spaces. In Sebastian Rudolph, Frithjof Dau, and Sergei O. Kuznetsov, editors, *ICCS*, volume 5662 of *Lecture Notes in Computer Science*, pages 173–184. Springer.

[Giesbrecht2010] Eugenie Giesbrecht. 2010. Towards a matrix-based distributional model of meaning. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Student Research Workshop*. ACL.

[Golan1992] Jonathan S. Golan. 1992. *The theory of semirings with applications in mathematics and theoretical computer science*. Addison-Wesley Longman Ltd.

[Grefenstette1994] Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Springer.

[Hopcroft and Ullman1979] John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[Kintsch2001] Walter Kintsch. 2001. Predication. *Cognitive Science*, 25:173–202.

[Lambek1958] Joachim Lambek. 1958. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170.

[Landauer and Dumais1997] Thomas K. Landauer and Susan T. Dumais. 1997. Solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, (104).

[Lund and Burgess1996] Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers*, 28:203–208.

[Mitchell and Lapata2008] Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244. ACL.

[Padó and Lapata2007] Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

[Plate1995] Tony Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

[Post1946] Emil L. Post. 1946. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268.

[Rendle et al.2009] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *KDD*, pages 727–736. ACM.

[Sahlgren et al.2008] Magnus Sahlgren, Anders Holst, and Pentti Kanerva. 2008. Permutations as a means to encode order in word space. In *Proc. CogSci'08*, pages 1300–1305.

[Salton et al.1975] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

[Schütze1993] Hinrich Schütze. 1993. Word space. In Lee C. Giles, Stephen J. Hanson, and Jack D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan-Kaufmann.

[Strang1993] Gilbert Strang. 1993. *Introduction to Linear Algebra*. Wellesley-Cambridge Press.

[Tucker1966] Ledyard R. Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3).

[Widdows2008] Dominic Widdows. 2008. Semantic vector products: some initial investigations. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*.