Structured Composition of Semantic Vectors

Stephen Wu
Mayo Clinic
wu.stephen@mayo.edu

William Schuler
The Ohio State University
schuler@ling.ohio-state.edu

Abstract

Distributed models of semantics assume that word meanings can be discovered from "the company they keep." Many such approaches learn semantics from large corpora, with each document considered to be unstructured bags of words, ignoring syntax and compositionality within a document. In contrast, this paper proposes a *structured* vectorial semantic framework, in which semantic vectors are defined and composed in syntactic context. As such, syntax and semantics are fully interactive; composition of semantic vectors necessarily produces a hypothetical syntactic parse. Evaluations show that using relationally-clustered headwords as a semantic space in this framework improves on a syntax-only model in perplexity and parsing accuracy.

1 Introduction

Distributed semantic representations like Latent Semantic Analysis (Deerwester et al., 1990), probabilistic LSA (Hofmann, 2001), Latent Dirichlet Allocation (Blei et al., 2003), or relational clustering (Taskar et al., 2001) have garnered widespread interest because of their ability to quantitatively capture 'gist' semantic content.

Two modeling assumptions underlie most of these models. First, the typical assumption is that words in the same document are an unstructured *bag of words*. This means that word order and syntactic structure are ignored in the resulting vectorial representations of meaning, and the only relevant relationship between words is the 'same-document' relationship. Second, these semantic models are not *compositional* in and of themselves. They require some external process to aggregate the meaning representations of words to form phrasal or sentential meaning; at best, they can jointly represent whole strings of words without the internal relationships.

This paper introduces *structured vectorial semantics* (SVS) as a principled response to these weaknesses of vector space models. In this framework, the syntax–semantics interface is fully interactive: semantic vectors exist in syntactic context, and any composition of semantic vectors necessarily produces a hypothetical syntactic parse. Since semantic information is used in syntactic disambiguation (MacDonald et al., 1994), we would expect practical improvements in parsing accuracy by accounting for the interactive interpretation process.

Others have incorporated syntactic information with vector-space semantics, challenging the bag-of-words assumption. Syntax and semantics may be jointly generated with Bayesian methods (Griffiths et al., 2005); syntactic structure may be coupled to the basis elements of a semantic space (Padó and Lapata, 2007); clustered semantics may be used as a pre-processing step (Koo et al., 2008); or, semantics may be learned in some defined syntactic context (Lin, 1998). These techniques are interactive, but their semantic models are not syntactically compositional (Frege, 1892). SVS is a generative model of sentences that uses a variant of the last strategy to incorporate syntax at preterminal tree nodes, but is inherently compositional.

Mitchell and Lapata (2008) provide a general framework for semantic vector composition:

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}, R, K) \tag{1}$$

where $\mathbf{u}$ and $\mathbf{v}$ are the vectors to be composed, R is syntactic context, K is a semantic knowledge base, and $\mathbf{p}$ is a resulting composed vector (or tensor). In this initial work of theirs, they leave out any notion of syntactic context, focusing on additive and multiplicative vector composition (with some variations):

$$\text{Add: } \mathbf{p}[i] = \mathbf{u}[i] + \mathbf{v}[i] \qquad \text{Mult: } \mathbf{p}[i] = \mathbf{u}[i] \cdot \mathbf{v}[i] \tag{2}$$

Since the structured vectorial semantics proposed here may be viewed within this framework, our discussion will begin from their definition in Section 2.1.

Erk and Padó's (2008) model also fits inside Mitchell and Lapata's framework, and like SVS, it includes syntactic context. Their semantic vectors use syntactic information as relations between multiple vectors in arriving at a final meaning representation. The emphasis, however, is on selectional preferences of individual words; resulting representations are similar to word-sense disambiguation output, and do not construct phrase-level meaning from word meaning. Mitchell and Lapata's more recent work (2009) combines syntactic parses with distributional semantics; but the underlying compositional model requires (as other existing models would) an interpolation of the vector composition results with a separate parser. It is thus not fully interactive.

Though the proposed structured vectorial semantics may be defined within Equation 1, the end output necessarily includes not only a semantic vector, but a full parse hypothesis. This slightly shifts the focus from the semantically-centered Equation 1 to an accounting of meaning that is necessarily interactive (between syntax and semantics); vector composition and parsing are then twin lenses by which the process may be viewed. Thus, unlike previous models, a unique *phrasal* vectorial semantic representation is composed during decoding.

Due to the novelty of phrasal vector semantics and lack of existing evaluative measures, we have chosen to report results on the well-understood dual problem of parsing. The structured vectorial semantic framework subsumes variants of several common parsing algorithms, two of which will be discussed: lexicalized parsing (Charniak, 1996; Collins, 1997, etc.) and relational clustering (akin to latent annotations (Matsuzaki et al., 2005; Petrov et al., 2006; Gesmundo et al., 2009)). Because previous work has shown that linguistically-motivated syntactic state-splitting already improves parses (Klein and Manning, 2003), syntactic states are split as thoroughly as possible into subcategorization classes (e.g., transitive and intransitive verbs). This pessimistically isolates the contribution of semantics on parsing accuracy — it will only show parsing gains where semantic information does not overlap with distributional syntactic information. Evaluations show that interactively considering semantic information with syntax has the predicted positive impact on parsing accuracy over syntax alone; it also lowers per-word perplexity.

The remainder of this paper is organized as follows: Section 2 describes SVS as both vector composition and parsing; Section 3 shows how relational-clustering SVS subsumes PCFG-LAs; and Section 4 evaluates modeling assumptions and empirical performance.

2 Structured Vectorial Semantics

2.1 Vector Composition

We begin with some notation. This paper will use boldfaced uppercase letters to indicate matrices (e.g., $\mathbf{L}$), boldfaced lowercase letters to indicate vectors (e.g., $\mathbf{e}$), and no boldface to indicate any single-valued variable (e.g. i). Indices of vectors and matrices will be associated with semantic concepts (e.g., $\mathrm{i}_1, \mathrm{i}_2, \ldots$); variables over those indices are single-value (scalar) variables (e.g., i); the contents of vectors and matrices can be accessed by index (e.g., $\mathbf{e}[\mathrm{i}_1]$ for a constant, $\mathbf{e}[i]$ for a variable). We will also define an operation $d(\cdot)$, which lists the elements of a column vector on the diagonal of a diagonal matrix, i.e., $d(\mathbf{e})[i, i]$=$\mathbf{e}[i]$. Often, these variables will technically be functions with arguments written in parentheses, producing vectors or matrices (e.g., $\mathbf{L}(l)$ produces a matrix based on the value of l).

As Mitchell and Lapata (2008) did, let us temporarily suspend discussion on what semantics populate our vectors for now. We can rewrite their equation (Equation 1) in SVS notation by following several conventions. All semantic vectors have a fixed dimensionality and are denoted $\mathbf{e}$; source vectors and the

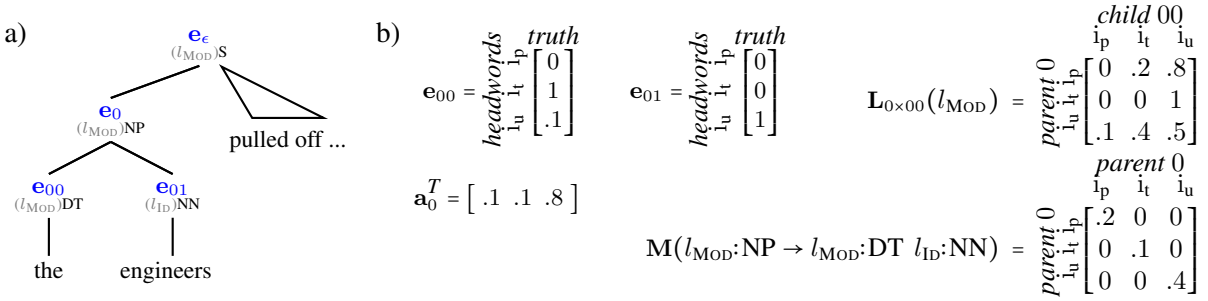

Figure 1: a) Syntax and semantics on a tree during decoding. Semantic vectors $\mathbf{e}$ are subscripted with the node's address. Relations l and syntactic categories c are constants for the example. b) Example vectors and matrices needed for the composition of a vector at address 0 (Section 2.2.1).

target vector are differentiated by subscript; instead of context variables R and K we will use M and L:

$$\mathbf{e}_\gamma = f(\mathbf{e}_\alpha, \mathbf{e}_\beta, M, L) \tag{3}$$

Syntactic context is in the form of grammar rules M that are aware of semantic concepts; semantic knowledge is in the form of labeled dependency relationships between semantic concepts, L. Both of these are present and explicitly modeled as matrices in SVS's canonical form of vector composition:

$$\mathbf{e}_\gamma = \mathbf{M} \cdot d(\mathbf{L}_{\gamma\times\alpha} \cdot \mathbf{e}_\alpha) \cdot d(\mathbf{L}_{\gamma\times\beta} \cdot \mathbf{e}_\beta) \cdot \mathbf{1} \tag{4}$$

Here, $\mathbf{M}$ is a diagonal matrix that encapsulates probabilistic syntactic information, where the syntactic probabilities depend on the semantic concept being considered. The $\mathbf{L}$ matrices are linear transformations that capture how semantically relevant source vectors are to the resulting vector (e.g., $\mathbf{L}_{\gamma\times\alpha}$ defines the the relevance of $\mathbf{e}_\alpha$ to $\mathbf{e}_\gamma$), with the intuition that two 1D vectors are under consideration and require a 2D matrix to relate them. $\mathbf{1}$ is a vector of ones — this takes a diagonal matrix and returns a column vector corresponding to the diagonal elements.

Of note in this definition of $f(\cdot)$ is the presence of matrices that operate on distributed semantic vectors. While it is widely understood that matrices can represent transformations, relatively few have used matrices to represent the distributed, dynamic nature of meaning composition (see Rudolph and Giesbrecht (2010) for a counterexample).

2.2 Syntax–Semantics Interface

This section aims to more thoroughly define the way in which the syntax and semantics interact during structured vectorial semantic composition. SVS will specify this interface such that the composition of semantic vectors is probabilistically consistent and subsumes parsing under various frameworks. Parsing has at times added semantic annotations that unwittingly carry some semantic value: headwords (Collins, 1997) are one-word concepts that subsume the words below them; latent annotations (Matsuzaki et al., 2005) are clustered concepts that touch on both syntactic and semantic information at a node. Of course, other annotations (Ge and Mooney, 2005) carry more explicit forms of semantics. In this light, semantic concepts (vector indices i) and relation labels (matrix arguments l) may also be seen as annotations on grammar trees.

Let us introduce notation to make the connection with parsing and syntax explicit. This paper will denote syntactic categories as c and string yields as x. The location of these variables in phrase structure will be identified using subscripts that describe the path from the root to the constituent.[1] Paths consist of left and/or right branches (indicated by '0's and '1's, respectively, as in Figure 1a). Variables α, β, and ι stand for whole paths; γ is the path of a composed vector; and ϵ is the empty path at the root. The yield x_γ is the observed (sub)string that eventually results from the progeny of c_γ. Multiple trees τ_γ can be constructed at γ by stringing together grammar rules that are consistent with observed text.

[1]For simplicity, trees are assumed to be compiled into strictly binary-branching form.

2.2.1 Lexicalized Parsing

To illustrate the definitions and operations presented in this section, we start with the concrete 'semantic' space of headwords (i.e., bilexical parsing) before moving on to a formal definition. Our example here corresponds to the best parse of the first two words in Figure 1a. In this example domain, assume that the semantic space of concept headwords is $\{\mathrm{i}_{\text{pulled}}, \mathrm{i}_{\text{the}}, \mathrm{i}_{\text{unk}}\}$, abbreviated as $\{\mathrm{i_p}, \mathrm{i_t}, \mathrm{i_u}\}$ where the last concept is a constant for infrequently-observed words. This semantic space becomes the indices of semantic vectors; complete vectors $\mathbf{e}$ at each node of Figure 1a are shown in Figure 1b.

The tree in Figure 1a contains complete concept vectors $\mathbf{e}$ at each node, with corresponding indices i. Values in these vectors (see Figure 1b) are probabilities, indicating the likelihood that a particular concept summarizes the meaning below a node. For example, consider $\mathbf{e}_{00}$: $\mathrm{i_t}$ produces the yield below address 00 ('the') with probability 1, and $\mathrm{i_u}$ may also produce 'the' with probability 0.1.

Not shown on the tree are the matrices in Figure 1b. In the parametrized matrix $\mathbf{M}(l_{\text{MOD}}{:}\text{NP} \rightarrow l_{\text{MOD}}{:}\text{DT}\ l_{\text{ID}}{:}\text{NN})$, each diagonal element corresponds to the hypothesized grammar rule's probability, given a headword. Similarly, the matrix $\mathbf{L}_{0\times 00}(l_{\text{MOD}})$ is parametrized by the semantic context l_{MOD} — here, l_{MOD} represents a generalized 'modifier' semantic role. For the semantic concept $\mathrm{i_p}$ at address 0, the left-child modifier (address 00) could be semantic concept $\mathrm{i_t}$ with probability 0.2, or concept $\mathrm{i_u}$ with probability 0.8. Finally, by adding an identity matrix for $\mathbf{L}_{0\times 01}(l_{\text{ID}})$ (a 'head' semantic role) to the quantities in Figure 1b, we would have all the components to construct the vector at address 0:

$$\mathbf{e}_0 = \underbrace{\begin{bmatrix} .2 & 0 & 0 \\ 0 & .1 & 0 \\ 0 & 0 & .4 \end{bmatrix}}_{\mathbf{M}} \cdot d\Bigg(\underbrace{\begin{bmatrix} 0 & .2 & .8 \\ 0 & 0 & 1 \\ .1 & .4 & .5 \end{bmatrix}}_{\mathbf{L}_{0\times 01}} \underbrace{\begin{bmatrix} 0 \\ 1 \\ .1 \end{bmatrix}}_{\mathbf{e}_{00}} \Bigg) \cdot d\Bigg(\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{L}_{0\times 01}} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{e}_{01}} \Bigg) \cdot \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}_{\mathbf{1}} = \begin{matrix} \\ \text{in} \\ \text{it} \\ \text{truth} \end{matrix} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0.036 \end{bmatrix}}_{\mathbf{e}_0}$$

Since the vector was constructed in syntactic and semantic context, the tree structure shown (including semantic relationships l) is implied by the context.

2.2.2 Probabilities in vectors and matrices

Formally defining the probabilities in Figure 1, SVS populates vectors and matrices by means of 5 probability models (models are denoted by θ), along with the process of composition:

$$\begin{aligned}
\text{Syntactic model} \qquad & \mathbf{M}(lc_\gamma \rightarrow lc_\alpha\ lc_\beta)[i_\gamma, i_\gamma] = \mathsf{P}_{\theta_{\text{M}}}(lci_\gamma \rightarrow lc_\alpha\, lc_\beta) \\
\text{Semantic model} \qquad & \mathbf{L}_{\gamma\times\iota}(l_\iota)[i_\gamma, i_\iota] = \mathsf{P}_{\theta_{\text{L}}}(i_\iota \mid i_\gamma, l_\iota) \\
\text{Preterminal model} \qquad & \mathbf{e}_\gamma[i_\gamma] = \mathsf{P}_{\theta_{\text{P-Vit(G)}}}(x_\gamma \mid lci_\gamma), \quad \text{for preterm } \gamma \\
\text{Root const. model} \qquad & \mathbf{a}_\epsilon^T[i_\epsilon] = \mathsf{P}_{\pi_{\text{G}\epsilon}}(lci_\epsilon) \\
\text{Any const. model} \qquad & \mathbf{a}_\gamma^T[i_\gamma] = \mathsf{P}_{\pi_{\text{G}}}(lci_\gamma)
\end{aligned} \tag{5}$$

These probabilities are encapsulated into vectors and matrices using a convention: column indices of vectors or matrices represent *conditioned* semantic variables, row indices represent *modeled* variables.

As an example, from Figure 1b, elements of $\mathbf{L}_{0\times 00}(l_{\text{MOD}})$ represent the probability $\mathsf{P}_{\theta_{\text{L}}}(i_{00} \mid i_0, l_{00})$. Thus, the conditioned variable i_{00} is shown in the figure as column indices, and the modeled i_0 as row indices. This convention applies to the $\mathbf{M}$ matrix as well. Recall that $\mathbf{M}$ is a diagonal matrix — its rows and columns model the same variable. Thus, we could rewrite $\mathsf{P}_{\theta_{\text{M}}}(lci_\gamma \rightarrow lc_\alpha\, lc_\beta)$ as $\mathsf{P}_{\theta_{\text{M}}}(lci_\gamma \rightarrow lc_\alpha\, lc_\beta, i_\gamma)$ to make a consistent probabilistic interpretation.

We have intentionally left out the probabilistic definition of normal (non-preterminal) nonterminals $\mathsf{P}_{\theta_{\text{Vit(G)}}}$, and the rationale for $\mathbf{a}^T$ vectors. These are both best understood in the dual problem of parsing.

2.2.3 Vector Composition for Parsing

The vector composition of Equation 4 can be rewritten with all arguments and syntactic information as:

$$\mathbf{e}_\gamma = \mathbf{M}(lc_\gamma \rightarrow lc_\alpha\ lc_\beta) \cdot d(\mathbf{L}_{\gamma\times\alpha}(l_\alpha) \cdot \mathbf{e}_\alpha) \cdot d(\mathbf{L}_{\gamma\times\beta}(l_\beta) \cdot \mathbf{e}_\beta) \cdot \mathbf{1} \tag{4'}$$

a compact representation that masks the underlying consistent probability operations. This section will expand the vector composition equation to show its equivalence to standard statistical parsing methods.

Let us say that $\mathbf{e}_\gamma[i_\gamma] = \mathsf{P}(x_\gamma \,|\, lci_\gamma)$, the probability of giving a particular yield given the present distributed semantics. Recall that in matrix multiplication, there is a summation over the inner dimensions of the multiplied objects; replacing matrices and vectors with their probabilistic interpretations and summing in the appropriate places, each element of $\mathbf{e}_\gamma$ is then:

$$\mathbf{e}_\gamma[i_\gamma] = \mathsf{P}_{\theta_\mathrm{M}}(lci_\gamma \rightarrow lc_\alpha\ lc_\beta) \cdot \sum_{i_\alpha} \mathsf{P}_{\theta_\mathrm{L}}(i_\alpha \,|\, i_\gamma, l_\alpha) \cdot \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_\alpha \,|\, lci_\alpha) \cdot \sum_{i_\beta} \mathsf{P}_{\theta_\mathrm{L}}(i_\beta \,|\, i_\gamma, l_\beta) \cdot \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_\beta \,|\, lci_\beta) \tag{6}$$

This can be loosely considered the multiplication of the syntax (θ_M term), left-child semantics (first sum), and right-child semantics (second sum). The only summations are between $\mathbf{L}$ and $\mathbf{e}$, since all other multiplications are between diagonal matrices (similar to pointwise multiplication).

We can simplify this probability expression by grouping θ_M and θ_L into a grammar rule $\mathsf{P}_{\theta_\mathrm{G}}(lci_\gamma \rightarrow lci_\alpha\ lci_\beta) \stackrel{\mathrm{def}}{=} \mathsf{P}_{\theta_\mathrm{M}}(lci_\gamma \rightarrow lc_\alpha\ lc_\beta) \cdot \mathsf{P}_{\theta_\mathrm{L}}(i_\alpha \,|\, i_\gamma, l_\alpha) \cdot \mathsf{P}_{\theta_\mathrm{L}}(i_\beta \,|\, i_\gamma, l_\beta)$, since they deal with everything except the yield of the two child nodes. The summations are then pushed to the front:

$$\mathbf{e}_\gamma[i_\gamma] = \sum_{i_\alpha, i_\beta} \mathsf{P}_{\theta_\mathrm{G}}(lci_\gamma \rightarrow lci_\alpha\ lci_\beta) \cdot \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_\alpha \,|\, lci_\alpha) \cdot \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_\beta \,|\, lci_\beta) \tag{7}$$

Thus, we have a standard chart-parsing probability $\mathsf{P}(x_\gamma \,|\, lci_\gamma)$ — with distributed semantic concepts — in each vector element.

The use of grammar rules necessarily builds a hypothetical subtree τ_γ. In a typical CKY algorithm, the tree corresponding to the highest probability would be chosen; however, we have not defined how to make this choice for vectorial semantics.

We will choose the best tree with probability 1.0, so we define a deterministic Viterbi probability over candidate *vectors* (not concepts) and context variables:

$$\mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_\gamma \,|\, lc\mathbf{e}_\gamma) \stackrel{\mathrm{def}}{=} [\![\, \mathbf{e}_\gamma = \arg\max_{lc\mathbf{e}_\iota} \left(\mathbf{a}_\iota^T \mathbf{e}_\iota \ \cdot \mathsf{P}_{\pi_\mathrm{G}}(lc\mathbf{a}_\iota^T) \cdot \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x \,|\, lc\mathbf{e}_\iota) \right)]\!] \tag{8}$$

where $[\![\cdot]\!]$ is an indicator function such that $[\![\phi]\!] = 1$ if ϕ is true, 0 otherwise. Intuitively, the process is as follows: we construct the vector $\mathbf{e}_\iota$ at a node, according to Eqn. 4′; we then weight this vector against prior knowledge about the context $\mathbf{a}_\iota^T$; the best vector in context will be chosen (the argmax). Also, the vector at a node comes with assumptions of what structure produced it. Thus, the last two terms in the parentheses are deterministic models ensuring that the best subtree τ_ι is indeed the one generated.

Determining the root constituent of the Viterbi tree is the same process as choosing any other Viterbi constituent, except that prior contextual knowledge gets its own probability model in $\mathbf{a}_\epsilon^T$. As before, the most likely tree $\hat{\tau}_\epsilon$ is the tree that maximizes the probability at the root, and can be constructed recursively from the best child trees. Importantly, $\hat{\tau}_\epsilon$ has an associated, *sentential* semantic vector which may be construed as the composed semantic information for the whole parsed sentence. Similar *phrasal* semantic vectors can be obtained anywhere on the parse chart.

These equations complete the linear algebraic definition of structured vectorial semantics.

3 SVS with Relational Clusters

3.1 Inducing Relational Clusters

Unlike many vector space models that are based on the frequencies of terms in documents, we may consider frequencies of terms that occur in similar semantic relations (e.g., head l_ID or modifier l_MOD). Reducing the dimensionality of terms in a term–context matrix will result in relationally-clustered concepts. From a parsing perspective, this amounts to latent annotations (Matsuzaki et al., 2005) in l-context.

Let us re-notate the headword-lexicalized version of SVS (the example in Section 2.2.1) using h for headword semantics, and reserve i for relationally-clustered concepts. Treebank trees can be deterministically annotated with headwords h and relations l by using head rules (Magerman, 1995). The 5 SVS models θ_{M}, θ_{L}, $\theta_{\text{P-Vit(G)}}$, $\pi_{\text{G}\epsilon}$, and π_{G} can thus be obtained by counting instances and normalizing. Empirical probabilities of this kind are denoted with a tilde, whereas estimated models have a hat.

Concepts i in a distributed semantic representation, however, cannot be found from annotated trees (see example concepts in Figure 2). Therefore, we use Expectation Maximization (EM) in a variant of the inside-outside algorithm (Baker, 1979) to learn distributed-concept behavior. In the M-step, the data-informed result of the E-step is used to update the estimates of θ_{M}, θ_{L}, and θ_{H} (where θ_{H} is a generlization of $\theta_{\text{P-Vit(G)}}$ to any nonterminal). These updated estimates are then plugged back in to the next E-step. The two steps continually alternate until convergence or a maximum number of iterations.

E-step:

$$\hat{\text{P}}(i_\gamma, i_\alpha, i_\beta \,|\, lc_\gamma, lc_\alpha, lc_\beta) = \frac{\hat{\text{P}}_{\theta_{\text{Out}}}(lci_\gamma, lch_\epsilon{-}lch_\gamma) \cdot \hat{\text{P}}_{\theta_{\text{Ins}}}(lch_\gamma \,|\, lci_\gamma)}{\hat{\text{P}}(lch_\epsilon)} \tag{9}$$

$$\text{E}(lci_\gamma, lci_\alpha, lci_\beta) = \hat{\text{P}}(i_\gamma, i_\alpha, i_\beta | lc_\gamma, lc_\alpha, lc_\beta) \cdot \tilde{\text{P}}(lc_\gamma, lc_\alpha, lc_\beta)$$

M-step:

$$\begin{aligned}
\hat{\text{P}}_{\theta_{\text{M}}}(lci_\gamma \rightarrow lc_\alpha, lc_\beta) &= \frac{\sum_{i_\alpha, i_\beta} \text{E}(lci_\gamma, lci_\alpha, lci_\beta)}{\sum_{lci_\alpha, lci_\beta} \text{E}(lci_\gamma, lci_\alpha, lci_\beta)} \\
\hat{\text{P}}_{\theta_{\text{L}}}(i_\alpha \,|\, i_\gamma; l_\alpha) &= \frac{\sum_{lc_\gamma, c_\alpha, lci_\beta} \text{E}(lci_\gamma, lci_\alpha, lci_\beta)}{\sum_{lc_\gamma, ci_\alpha, lci_\beta} \text{E}(lci_\gamma, lci_\alpha, lci_\beta)} \\
\hat{\text{P}}_{\theta_{\text{H}}}(h_\gamma \,|\, lci_\gamma) &= \frac{\text{E}(lci_\gamma, -, -)}{\sum_{h_\gamma} \text{E}(lci_\gamma, -, -)}
\end{aligned} \tag{10}$$

Inside probabilities can be recursively calculated on training trees from the bottom up. These are simply probability sums of all subsumed subtrees (Viterbi probabilities with sums instead of maxes).

Outside probabilities can also be recursively calculated from training trees, here from parent probabilities. For a left child (the right-child case is similar):

$$\begin{aligned}
\hat{\text{P}}_{\theta_{\text{Out}}}(lci_\alpha, lch_\epsilon{-}lch_\alpha) = &\hat{\text{P}}_{\theta_{\text{Out}}}(lci_\gamma, lch_\epsilon{-}lch_\gamma) \cdot \hat{\text{P}}_{\theta_{\text{M}}}(lci_\gamma \rightarrow lc_\alpha, lc_\beta) \\
&\cdot \textstyle\sum_{i_\beta} \hat{\text{P}}_{\theta_{\text{L}}}(i_\beta \,|\, i_\gamma, l_\beta) \cdot \hat{\text{P}}_{\theta_{\text{Ins}}}(lch_\beta \,|\, lci_\beta) \cdot \hat{\text{P}}_{\theta_{\text{L}}}(i_\alpha \,|\, i_\gamma, l_\alpha)
\end{aligned} \tag{11}$$

Since outside probabilities signify everything *but* what is subsumed by the node, they carry a complementary set of information to inside probabilities. Thus, inside and outside probabilities together are a natural way to produce parent and child clustered concepts.

3.2 Relational Semantic Clusters in Parsing

Section 2.2.2 listed the five probability models necessary for SVS. To define SVS with relational clusters, the estimates in Equation 10 can be used for θ_{M} and θ_{L}.

The preterminal model is based on θ_{H}, but it also includes some backoff for words that have not been used as headwords. The other two models also fall out nicely from the algorithm, though they are not explicitly estimated in EM. The prior probability at the root is just the base case for outside probabilities:

$$\hat{\text{P}}_{\pi_{\text{G}\epsilon}}(lci_\epsilon) \stackrel{\text{def}}{=} \hat{\text{P}}_{\theta_{\text{Out}}}(lci_\epsilon, lch_\epsilon{-}lch_\epsilon) \tag{12}$$

Prior probabilities at non-root constituents are estimated from the empirically-weighted joint probability.

$$\hat{\text{P}}_{\pi_{\text{G}}}(lci_\gamma) \stackrel{\text{def}}{=} \sum_{lci_\alpha, lci_\beta} \tilde{\text{P}}(lci_\gamma, lci_\alpha, lci_\beta) \tag{13}$$

With these models, a relationally-clustered SVS parser is now defined.

Cluster i_1 'announcement'		Cluster i_5 'change in value'		Cluster i_7 'change possession'	
unk	0.362	rose	0.137	unk	0.381
was	0.173	fell	0.124	had	0.065
reported	0.097	unk	0.116	was	0.062
posted	0.036	gained	0.063	took	0.036
earned	0.029	dropped	0.051	bought	0.027
filed	0.024	attributed	0.051	completed	0.025
were	0.022	jumped	0.046	received	0.024
had	0.020	added	0.041	were	0.023
told	0.013	lost	0.039	got	0.018
approved	0.013	advanced	0.022	made	0.018
				acquired	0.016

Figure 2: Example θ_{H} clusters from 1,000 headwords clustered into 10 referents, after 10 EM iterations, for transitive past-tense verbs (VBD-argNP).

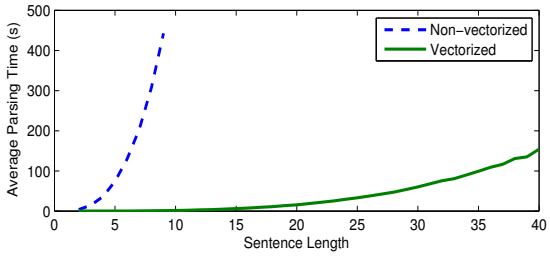

Figure 3: Speed of relationally-clustered SVS parsers, with and without vectorization.

4 Evaluation

Sections 02–21 of the Wall Street Journal (WSJ) corpus were used as training data; Section 23 was used as test data with reported parsing results on sentences greater than length 40. Punctuation was left in for all reported evaluations. Trees were binarized, and syntactic states were thoroughly split into subcategorization classes. As previously discussed, unlike tests on state-of-the-art automatically state-splitting parsers, this isolates the contribution of semantics. The baseline 83.57 F-measure is comparable to Klein and Manning (2003) before the inclusion of head annotations.

Subsequently, each branch was annotated with a head relation l_{ID} or a modifier relation l_{MOD} according to a binarized version of headword percolation rules (Magerman, 1995; Collins, 1997), and the headword was propagated up from its head constituent. The most frequent headwords (e.g., $h_1, \ldots, h_{50}$) were stored, and the rest were assigned a constant, 'unk' headword category.

From counts on the binary rules of these annotated trees, the θ_{M}, θ_{L}, $\theta_{\text{P-Vit(G)}}$, $\pi_{\text{G}\epsilon}$, and π_{G} probabilities for headword-lexicalization SVS were obtained. Modifier relations l_{MOD} were deterministically augmented with their syntactic context; both c and l symbols appearing fewer than 10 times in the whole corpus were assigned 'unknown' categories.

These lexicalized models served as a baseline, but the augmented trees from which they were derived were also inputs to the EM algorithm in Section 3.1. Each parameter in the model or training algorithm was examined, with $|I| = \{1, 5, 10, 15, 20\}$ clusters, random initialization from reproducible seeds, and a varying numbers of EM iterations.

The implemented parser had few adjustments from a plain CKY parser other than these vectors. No approximate inference was used, with no beam for candidate parses and no re-ranking.

4.1 Interpretable relational clusters

Figure 2 shows example clusters for one of the headword models used, where EM clustered 1,000 headwords into 10 concepts in 10 iterations. The lists are parts of the $\hat{\text{P}}_{\theta_{\text{H}}}(h_\gamma \,|\, lci_\gamma)$ model. As such, each of the 10 clusters will only produce headwords in light of some syntactic constituent. The figure shows how distributed concepts produce headwords for transitive past-tense verbs. Note that the probability distributions for different headwords are quite uneven, again confirming that some clusters are more specific, and others are more general.

Each cluster has been given a heading of its approximate meaning — i_5, for example, mostly picks verbs that are 'change in value' events. With 10 clusters, we might not expect such fine-grained clusters, since pLSA-related approaches typically use several hundred for such tasks. The syntactic context of transitive (and therefore state-split) past-tense verbs allows for much finer-grained distinctions, which are then predominantly semantic in nature.

a)

Sec. 23, length < 40 wds	LR	LP	F
syntax-only baseline:	83.32	83.83	83.57
headword-lex. 10hw:	83.10	83.61	83.35
headword-lex. 50hw:	83.09	83.40	83.24
rel'n clust. 50hw→10 clust:	83.67	84.13	**83.90**

b)

Sec. 23, length < 40 wds	LR	LP	F
baseline→1 clust	83.34	83.90	83.62
1000 hw→5 clust, avg	83.85	84.23	84.04
1000 hw→10 clust, avg	84.04	84.40	84.21
1000 hw→15 clust, avg	84.15	84.38	84.26
1000 hw→20 clust, avg	**84.21**	**84.42**	**84.31**

Table 1: a) Unsmoothed lexicalized CKY parsers versus 10 semantic clusters. Evaluations were run with EM trained to 10 iterations. b) Average dependence of parsing performance on number of semantic clusters. Averages are taken over different random seeds, with EM running 4 or 10 iterations.

4.2 Engineering considerations

We should note that relationally-clustered SVS is feasible with respect to random initialization and speed.

Four relationally-clustered SVS models (with 500 headwords clustered into 5 concepts) were trained, each having a different random initialization. We found that the parsing F-score had a mean of 83.98 and a standard deviation of 0.21 across different initializations of the model. This indicates that though there are significant difference between the models, they still outperform models without SVS (see next section).

Also, it may seem slow to consider the set of semantic concepts and relations alongside syntax, at least with respect to normal parsing. The definition of SVS in terms of vectors actually mitigates this effect on WSJ Section 23, according to Figure 3. Since SVS is probabilistically consistent, the parser could be defined without vectors, but this would have the 'non-vectorized' speed curve. The contiguous storage and access of information in the 'vectorized' version leads to an efficient implementation.

4.3 Comparison to Lexicalization

One important comparison to draw here is between the effectiveness of semantic clusters versus headword-lexicalization. For fair head-to-head comparison on WSJ Section 23, both models were vectorized and included no smoothing or backoff. Neither relational clusters nor lexicalization were optimized with backoff or smoothing.

Table 1a shows precision, recall, and F-score for lexicalized models and for clustered semantic models. First, note that the 10-cluster model (in bold) improves on a syntax-only parser (top line), showing that the semantic model is contributing useful information to the parsing task.

Next, compare the 50-headword, 10-cluster model (in bold) to the line above it. It is natural to compare this model to the headword-lexicalized model with 50 headwords, since the same information from the trees is available to both models. The relationally-clustered model outperforms the headword-lexicalized model, showing that clustering the headwords actually improves their usefulness, despite the fact that fewer referents are used in the actual vectors.

It is also interesting, then, to compare this 50-headword, 10-cluster model to a headword-lexicalized model with 10 headwords. In this case, the possible size of the grammar is equal. Again, the relationally-clustered model outperforms plain lexicalization. This indicates that the 10 clustered referents are much more meaningful than 10 headword referents for the disambiguating of syntax.

4.4 Effect of Number of clusters

The final experiment on relational-clustering SVS was to determine whether performance would vary with the number of clusters. Table 1b compares average performance (over different random initializations) for numbers of clusters from 1 (a syntax-equivalent case) to 20.

First, it should be noted that all of the relationally clustered models improved on the baseline. Random initializations did not vary enough for these models to do worse than syntax alone. For each vector/domain size, in fact, the gains over syntax-only are substantial.

In addition, the table shows that average performance increases with the number of clusters. This loosely positive slope means that EM is still finding useful parts of the semantic space to explore and cluster, so that the clusters remain meaningful. However, the increase in performance with number of clusters is likely to eventually plateau.

Maximum-accuracy models were also evaluated, since each model is a full-fledged parser. The best 20-referent model obtained an F score of 84.60%, beating the syntactic baseline by almost a full absolute point. Thus, finding relationally-clustered semantic output also contributes to some significant parsing benefit.

4.5 Perplexity

Finally, per-word perplexities were calculated for a syntactic model and for a 5-concept relationally-clustered model. Specific to this evaluation, following Mitchell and Lapata (2009), only the top 20,000 words in WSJ Sections 02-21 were kept in training or test sentences, and the rest replaced with 'unk'; numbers were replaced with 'num.'

Sec. 23, 'unk'+'num'	Perplexity
syntax only baseline	428.94
rel'n clust. 1khw→005e	371.76

Table 2: Model fit as measured by perplexity.

Table 2 shows that adding semantic information greatly reduces perplexity. Since as much syntactic information as possible (such as argument structure) has been pre-annotated onto trees, the isolated contribution of interactive semantics improves on a syntax-only model model.

5 Conclusion

This paper has introduced a structured vectorial semantic (SVS) framework in which vector composition and syntactic parsing are a single, interactive process. The framework thus fully integrates distributional semantics with traditional syntactic models of language.

Two standard parsing techniques were defined within SVS and evaluated: headword-lexicalization SVS (bilexical parsing) and relational-clustering SVS (latent annotations). It was found that relationally-clustered SVS outperformed the simpler lexicalized model and syntax-only models, and that additional clusters had a mildly positive effect. Additionally, perplexity results showed that the integration of distributed semantics in relationally-clustered SVS improved the model over a non-interactive baseline.

It is hoped that this flexible framework will enable new generations of interactive interpretation models that deal with the syntax–semantics interface in a plausible manner.

References

Baker, J. (1979). Trainable grammars for speech recognition. In D. Klatt and J. Wolf (Eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003). Latent dirichlet allocation. *Journal of Machine Learning Research 3*, 993–1022.

Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 1031–1036.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL '97)*.

Deerwester, S., S. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*(6), 391–407.

Erk, K. and S. Padó (2008). A structured vector space model for word meaning in context. In *Proceedings of EMNLP 2008*.

Frege, G. (1892). Uber sinn und bedeutung. *Zeitschrift fur Philosophie und Philosophischekritik 100*, 25–50.

Ge, R. and R. J. Mooney (2005). A statistical semantic parser that integrates syntax and semantics. In *Ninth Conference on Computational Natural Language Learning*, pp. 9–16.

Gesmundo, A., J. Henderson, P. Merlo, and I. Titov (2009). A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of CoNLL*, pp. 37–42. Association for Computational Linguistics.

Griffiths, T. L., M. Steyvers, D. M. Blei, and J. B. Tenenbaum (2005). Integrating topics and syntax. *Advances in neural information processing systems 17*, 537–544.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning 42*(1), 177–196.

Klein, D. and C. D. Manning (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, pp. 423–430.

Koo, T., X. Carreras, and M. Collins (2008). Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the ACL*, Volume 8. Citeseer.

Lin, D. (1998). An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, Volume 296304.

MacDonald, M. C., N. J. Pearlmutter, and M. S. Seidenberg (1994). The lexical nature of syntactic ambiguity resolution. *Psychological Review 101*(4), 676–703.

Magerman, D. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, Cambridge, MA, pp. 276–283.

Matsuzaki, T., Y. Miyao, and J. Tsujii (2005). Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 75–82. Association for Computational Linguistics.

Mitchell, J. and M. Lapata (2008). Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, Columbus, OH, pp. 236–244.

Mitchell, J. and M. Lapata (2009). Language Models Based on Semantic Composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 430–439.

Padó, S. and M. Lapata (2007). Dependency-based construction of semantic space models. *Computational Linguistics 33*(2), 161–199.

Petrov, S., L. Barrett, R. Thibaux, and D. Klein (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'06)*.

Rudolph, S. and E. Giesbrecht (2010). Compositional matrix-space models of language. In *Proceedings of ACL 2010*. Association for Computational Linguistics.

Taskar, B., E. Segal, and D. Koller (2001). Probabilistic classification and clustering in relational data. In *IJCAI'01: Proceedings of the 17th international joint conference on Artificial intelligence*, San Francisco, CA, USA, pp. 870–876. Morgan Kaufmann Publishers Inc.