Structured Vectorial Semantics for Broad-coverage Parsing

Abstract

This paper proposes a novel structured vectorial semantic framework, incorporating cognitively-motivated 'gist' semantics as selectional restrictions in interactive vectorial-semantic parsing. Applying vectorial semantic techniques (in particular, relational clustering over dependent headwords) in this extended framework has a predictable positive impact on parsing accuracy. The vectorial representation is conducive to a fast implementation of necessary parser operations, despite the fact that vectorial semantic techniques have dense relations to propagate through the parse chart.

1 Introduction

Vectorial semantic representations like Latent Semantic Analysis (Deerwester et al., 1990), probabilistic LSA (Hofmann, 1999; Hofmann, 2001), Latent Dirichlet Allocation (Blei et al., 2003), relational clustering (Taskar et al., 2001), or tensor clustering (Banerjee et al., 2007) provide a means of exploiting hidden or 'latent' semantics. These kinds of models have been described as 'gist semantics' and likened to the associative properties of human memory (Landauer and Dumais, 1997; Kintsch, 2001). Vector semantic spaces are typically based on co-occurrences between words (Lund and Burgess, 1996) in one or more relations, expressed as matrices, and employed for the purpose of improving accuracy in document search.

This paper proposes a novel *structured* vectorial semantics, incorporating cognitively-motivated 'gist' semantics as selectional restrictions in an interactive vectorial-semantic parsing framework. Vectorial semantics are used in a generative parser by factoring an augmented PCFG model into (roughly) syntactic components and semantic components. Both components are encapsulated in vectors and matrices; they are recursively composed using matrix multiplication and diagonal listing.

Vectorial semantics are expected interact constructively with syntax because they leverage observed robust semantic information in the recognition task. Word co-occurrence matrices are reduced in dimensionality to capture their most salient information in fewer, denser concepts. Relationships between these denser concepts are similar in meaning to relationships between the words from which they were derived, but they are less prone to data sparsity. Thus, including such semantic relationships alongside syntactic structure provides a robust, complimentary resource in an interpretation framework.

Typically, vectorial semantics techniques model documents as unstructured bags of words, but there has been recent interest in considering syntactic information alongside semantics (Griffiths et al., 2005; Boyd-Graber and Blei, 2008; Padó and Lapata, 2007; Erk and Padó, 2008), even synthesizing vectorial semantics with parsing (Mitchell and Lapata, 2009). Extending or using these models for broad-coverage parsing requires an interpolation of the results from an independent semantic composer and from a syntactic parser. In contrast, the proposed model fully interleaves vectorial semantic composition with the recursive construction of phrase structure in parsing.

Two preliminary instantiations of the structured vectorial semantic framework are presented and evaluated: lexicalized parsing (Collins, 1997; Eisner and Satta, 1999, etc.) and relational clustering (akin to latent annotations (Matsuzaki et al., 2005; Petrov et al., 2006)).

Vectorial semantic techniques are applied in the latter of these, yielding a predictable positive impact on parsing accuracy. In addition, the vectorial representation is conducive to a fast implementation of necessary parser operations; this is despite the fact that vectorial semantic techniques have dense, dimensionality-reduced relations to consider and propagate through the full syntactic structure of hypothesized parses. It is hoped that this plausible, flexible framework will enable new generations of cognitive models and evaluations.

The remainder of this paper is organized as follows: Section 2 describes the theoretical framework; Section 3 instantiates the framework with relational clusters; and Section 4 evaluates modeling assumptions and parsing performance.

2 Structured Vectorial Semantics

2.1 Syntactic Parsing

The parsing framework for including semantic clusters is based on standard probabilistic context-free grammars. 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, or variables ι), concatenated into sequences η, and ϵ is the empty sequence at the root. The yield x_η is the observed (sub)string which eventually results from the progeny of c_η. Multiple trees τ_η can be constructed at η by stringing together grammar rules that are consistent with observed text.

For a phrase structure tree rooted at a constituent of category c_η with yield x_η, the task of parsing will require the calculation of the Viterbi (best) tree and substring probability. These can be calculated using probabilities assigned to grammar rules $\mathsf{P}_{\theta_\mathrm{G}}(c_\eta \to c_{\eta 0}\, c_{\eta 1})$ in a probabilistic context-free grammar (PCFG) model θ_G.

Any yield x_η can be decomposed into (assembled from) prefix $x_{\eta 0}$ and suffix $x_{\eta 1}$ yields, i.e., $x_\eta = x_{\eta 0}\, x_{\eta 1}$. Viterbi scores (probability of the best tree) maximize over such decompositions:

$$\mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_\eta \,|\, c_\eta) = \max_{x_{\eta 0} c_{\eta 0}, x_{\eta 1} c_{\eta 1}} \mathsf{P}_{\theta_\mathrm{G}}(c_\eta \to c_{\eta 0}\; c_{\eta 1}) \cdot \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_{\eta 0} \,|\, c_{\eta 0}) \cdot \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x_{\eta 1} \,|\, c_{\eta 1}) \quad (1)$$

and the corresponding tree $\hat{\tau}_\eta$ can be constructed from best child trees $\hat{\tau}_{\eta 0}$ and $\hat{\tau}_{\eta 1}$.

Given some prior model $\mathsf{P}_{\pi_\mathrm{G}}(c_\epsilon)$, the probability at the root node can be obtained:

$$\mathsf{P}(x\, c_\epsilon) = \mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x \,|\, c_\epsilon) \cdot \mathsf{P}_{\pi_\mathrm{G}}(c_\epsilon) \quad (2)$$

The Viterbi probability $\mathsf{P}_{\theta_{\mathrm{Vit(G)}}}(x \,|\, c_\epsilon)$ that maximizes this root probability will have an associated tree $\hat{\tau}_\epsilon$ that includes the root syntactic category c_ϵ; this tree can be constructed by referring back to best subtrees at its children, $\hat{\tau}_0$ and $\hat{\tau}_1$, as in a standard CKY algorithm.

2.2 Syntactic–Semantic Parsing

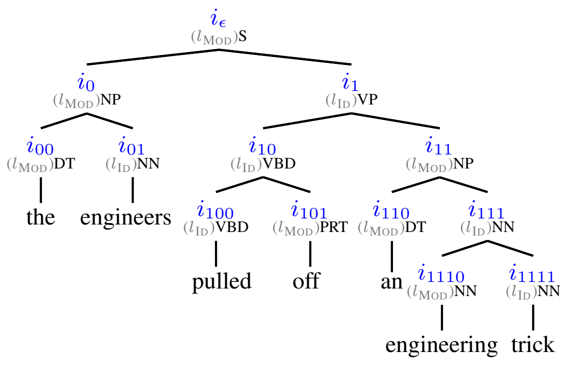
In grammatically-structured semantics, both semantic *concepts* (written i) and *relations* (written l) between these concepts are tied to grammatical structure. The task, then, is to *jointly* determine what syntax and semantics best match the observed input.

Figure 1 shows an example of a tree that has concepts i, relations l, and syntactic categories c annotated. A probabilistic grammar producing this figure would expand differently for each different concept i_η in a domain of concepts E.

Instead of directly generating joint lci_η probabilities (collocated variables are indexed together for clarity), such a syntactic–semantic grammar rule can be factored into (loosely) a syntactic component and a semantic component.

$$\begin{aligned}\mathsf{P}_{\theta_\mathrm{G}}(lci_\eta \to lci_{\eta 0}\; lci_{\eta 1}) = \\ \mathsf{P}_{\theta_\mathrm{M}}(lci_\eta \to lc_{\eta 0}\; lc_{\eta 1}) \\ \cdot \mathsf{P}_{\theta_\mathrm{L}}(i_{\eta 0} \,|\, i_\eta; l_{\eta 0}) \cdot \mathsf{P}_{\theta_\mathrm{L}}(i_{\eta 1} \,|\, i_\eta; l_{\eta 1}) \quad (3)\end{aligned}$$

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



i_ϵ (l_{MOD})S
i_0 (l_{MOD})NP
i_1 (l_{ID})VP
i_{00} (l_{MOD})DT
i_{01} (l_{ID})NN
i_{10} (l_{ID})VBD
i_{11} (l_{MOD})NP
the
engineers
i_{100} (l_{ID})VBD
i_{101} (l_{MOD})PRT
i_{110} (l_{MOD})DT
i_{111} (l_{ID})NN
pulled
off
an
i_{1110} (l_{MOD})NN
i_{1111} (l_{ID})NN
engineering
trick

Figure 1: Syntax and semantics annotated on a tree. Concepts i are subscripted with the node's address. Relations l and syntactic categories c are specific to the example.

In the θ_{M} model, concepts constrain the generation of syntax and of acceptable relations for a child constituent. In the θ_{L} models, child concepts are probabilistically connected to parent concepts on the basis of the θ_{M}-generated child relations. The semicolon in $\mathsf{P}_{\theta_{\text{L}}}(i_{\eta\iota} \,|\, i_\eta; l_{\eta\iota})$ is used to show that the relationship is between parent and child concepts, but that the probability is parameterized by (and thus conditioned on) l_η.

Viterbi probabilities from Eqn. 1 are extended by marginalizing out child concepts $i_{\eta 0}$ and $i_{\eta 1}$.

$$\begin{aligned}\mathsf{P}_{\theta_{\text{Vit(G)}}}(x_\eta \,|\, lci_\eta) = \\ \max_{xlc_{\eta 0}, xlc_{\eta 1}} \sum_{i_{\eta 0}, i_{\eta 1}} \mathsf{P}_{\theta_{\text{G}}}(lci_\eta \rightarrow lci_{\eta 0}\; lci_{\eta 1}) \\ \cdot \mathsf{P}_{\theta_{\text{Vit(G)}}}(x_{\eta 0} \,|\, lci_{\eta 0}) \cdot \mathsf{P}_{\theta_{\text{Vit(G)}}}(x_{\eta 1} \,|\, lci_{\eta 1}) \quad (4)\end{aligned}$$

A possible alternative would be to take the maximum over $i_{\eta 0}$ and $i_{\eta 1}$. Grammar rule probabilities from θ_{G} are factored as in Eqn. 3.

Then, given a prior probability $\mathsf{P}_{\pi_{\text{G}\epsilon}}(lci_\epsilon)$, the probability at the root is:

$$\mathsf{P}(x\, lc_\epsilon) = \sum_{i_\epsilon} \mathsf{P}_{\theta_{\text{Vit(G)}}}(x_\epsilon \,|\, lci_\epsilon) \cdot \mathsf{P}_{\pi_{\text{G}\epsilon}}(lci_\epsilon) \quad (5)$$

where again, the maximum root probability has a Viterbi probability with corresponding most likely tree $\hat{\tau}_\epsilon$.

One obvious reason to factor into θ_{M} and θ_{L} is to reduce the sparsity of training data: syntactic grammars have potentially $|C|^3$ rules, whereas unfactored structured semantic rules could have $(|C||L||E|)^3$ rules.

But the crucial point is that models without this factorization (Matsuzaki et al., 2005; Petrov et al., 2006) will be unable to maintain a cohesive semantic representation across all syntactic constituents. Such strategies have proven useful for parsing, where the only output of concern is a parse tree (or qualitative analyses of the state-split constituents). However, all other uses for vector-space semantics, such as judging document (or sentence, or phrase, or word) similarity, are precluded from non-cohesive models.

2.3 Structured Vectorial Semantic Parsing

A framework is now needed by which grammatically-structured semantics can be cast as vectors and parsed. For structured vectorial semantics, a vector or matrix will represent probabilities in a semantic space. Notation and definitions are provided here for encapsulating probability distributions *in* vectors, and for defining parsing probabilities *over* those vectors.

This paper will use boldfaced uppercase letters to indicate matrices (e.g., relations $\mathbf{L}$), boldfaced lowercase letters to indicate vectors (e.g., concepts $\mathbf{e}$), and no boldface to indicate any single-valued variable (e.g. labels l that indicate which $\mathbf{L}$ to use). Often, these variables will technically be functions with arguments written in parentheses, producing the expected vectors or matrices (e.g., $\mathbf{L}(l_\eta)$ will produce some matrix based on the argument l_η).

Indices of vectors and matrices will be associated with semantic concepts (e.g., $\mathrm{i}_1, \ldots, \mathrm{i}_{|\mathbf{e}|}$); variables over those indices are single-value (scalar) variables (e.g., i_η). The contents of vectors and matrices can also be accessed by index (e.g., $\mathbf{e}[\mathrm{i}_1]$ for a constant, $\mathbf{e}[i]$ for a variable). This article will adopt the convention that column indices of vectors or matrices represent values of conditioned semantic variables, and row indices represent values of modeled variables.

A running example corresponding to the best parse of the first two words of Figure 1 will illustrate the definitions and operations presented in this section, using the concrete semantic space of headwords.[2] In this example do-

[2]Headwords are usually considered a syntactic feature,

main, assume that the space of concept headwords is $E = \{\mathrm{h}_{\text{pulled}}, \mathrm{h}_{\text{the}}, \mathrm{h}_{\text{unk}}\}$, abbreviated as $\{\mathrm{h_p}, \mathrm{h_t}, \mathrm{h_u}\}$ where the last concept is a constant for infrequently-observed words.

First, per-concept Viterbi probabilities in a syntactic context lc_η are encapsulated as column vectors:

$$\mathbf{e}_\eta = i_\eta \mapsto \mathsf{P}_{\theta_{\text{Vit(G)}}}(x \,|\, lci_\eta) \tag{6}$$

where $i_\eta \mapsto \cdots$ indicates that what follows is an element-by-element definition of $\mathbf{e}_\eta$. So at the level of word generation from address 0 in the example, the left child concept will be assigned from a special preterminal Viterbi probability $\mathsf{P}_{\theta_{\text{P-Vit(G)}}}(\text{the} \,|\, l_{\text{MOD}}\!:\!\text{DT}\{\mathrm{h_t}\})$, and the right child will be from $\mathsf{P}_{\theta_{\text{P-Vit(G)}}}(\text{egrs.} \,|\, l_{\text{ID}}\!:\!\text{NN}\{\mathrm{h_u}\})$:

$$\mathbf{e}_{00} = \begin{array}{c|c} & \textit{truth} \\ \textit{headwords}\ \mathrm{h_p} & 0 \\ \mathrm{h_t} & 1 \\ \mathrm{h_u} & .1 \end{array} \qquad \mathbf{e}_{01} = \begin{array}{c|c} & \textit{truth} \\ \textit{headwords}\ \mathrm{h_p} & 0 \\ \mathrm{h_t} & 0 \\ \mathrm{h_u} & 1 \end{array} \tag{6$'$}$$

Next, grammar probabilities are encapsulated per-concept in the $\mathbf{M}$ matrix. There is a different diagonal matrix for each binary rule:

$$\mathbf{m}(lc_\eta \to lc_{\eta0}\ lc_{\eta1}) \overset{\text{def}}{=} i_\eta \mapsto \mathsf{P}_{\theta_{\text{M}}}(lci_\eta \to lc_{\eta0}\ lc_{\eta1})$$
$$\mathbf{M}(lc_\eta \to lc_{\eta0}\ lc_{\eta1}) \overset{\text{def}}{=} \mathrm{d}(\mathbf{m}(lc_\eta \to lc_{\eta0}\ lc_{\eta1})) \tag{7}$$

where $\mathrm{d}(\cdot)$ lists the elements of a column vector on the diagonal of a diagonal matrix. For our example, assume the following rule expansion probabilities:

$$\mathbf{M}(l_{\text{MOD}}\!:\!\text{NP} \to l_{\text{MOD}}\!:\!\text{DT}\ l_{\text{ID}}\!:\!\text{NN}) = \begin{array}{c|ccc} & \multicolumn{3}{c}{\textit{parent } \eta} \\ & \mathrm{h_p} & \mathrm{h_t} & \mathrm{h_u} \\ \textit{parent } \eta\ \mathrm{h_p} & .2 & 0 & 0 \\ \mathrm{h_t} & 0 & .1 & 0 \\ \mathrm{h_u} & 0 & 0 & .4 \end{array} \tag{7$'$}$$

Relation matrices $\mathbf{L}$ are functions from parent concepts i_η to child concepts $i_{\eta0}$ or $i_{\eta1}$.

$$\mathbf{L}(l_{\eta\iota}) = \mathbf{L}_{\eta\times\eta\iota}(l_{\eta\iota}) \overset{\text{def}}{=} i_\eta \mapsto i_{\eta\iota} \mapsto \mathsf{P}_{\theta_{\text{L}}}(i_{\eta\iota} \,|\, i_\eta; l_{\eta\iota}) \tag{8}$$

In the example, a nonterminal constituent will share its headword with one of its sub-constituents (its head), so this sub-constituent will relate to its parent via the identity matrix $\mathbf{L}(l_{\text{ID}}) = \mathbf{I}$. The other sub-constituent will use a general modifier relation[3] $\mathbf{L}(l_{\text{MOD}})$. The modifier relation may plausibly be:

$$\mathbf{L}_{0\times00}(l_{\text{MOD}}) = \begin{array}{c|ccc} & \multicolumn{3}{c}{\textit{child } 00} \\ & \mathrm{h_p} & \mathrm{h_t} & \mathrm{h_u} \\ \textit{parent } 0\ \mathrm{h_p} & 0 & .2 & .8 \\ \mathrm{h_t} & 0 & 0 & 1 \\ \mathrm{h_u} & .1 & .4 & .5 \end{array} \tag{8$'$}$$

Next, the prior probability of a constituent is a row vector, distributed over i_η:

$$\mathbf{a}_\eta^T = i_\eta \mapsto \mathsf{P}_{\pi_{\text{G}}}(lci_\eta) \tag{9}$$

This is the prior probability of any constituent in the grammar, similar to the root constituent prior probability $\pi_{\text{G}\epsilon}$. As an example of the prior probability of an NP with each concept:

$$\mathbf{a}_0^T = \begin{bmatrix} .1 & .1 & .8 \end{bmatrix} \tag{9$'$}$$

Combining Eqn. 6 (and 4) with these vector and matrix encapsulations of probability distributions, parsing probabilities can be encapsulated in a vector:

$$\begin{aligned} \mathbf{e}_\eta &= i_\eta \mapsto \sum_{i_{\eta0}, i_{\eta1}} \mathsf{P}_{\theta_{\text{G}}}(lci_\eta \to lci_{\eta0}\ lci_{\eta1}) \\ &\qquad \cdot \mathsf{P}_{\theta_{\text{Vit(G)}}}(x_{\eta0} \,|\, lci_{\eta0}) \cdot \mathsf{P}_{\theta_{\text{Vit(G)}}}(x_{\eta1} \,|\, lci_{\eta1}) \\ &= i_\eta \mapsto \mathsf{P}_{\theta_{\text{M}}}(lci_\eta \to lc_{\eta0}\ lc_{\eta1}) \\ &\quad \cdot \sum_{i_{\eta0}} \mathsf{P}_{\theta_{\text{L}}}(i_{\eta0} \,|\, i_\eta; l_{\eta0}) \cdot \mathsf{P}_{\theta_{\text{Vit(G)}}}(x_{\eta0} \,|\, lci_{\eta0}) \\ &\qquad \cdot \sum_{i_{\eta1}} \mathsf{P}_{\theta_{\text{L}}}(i_{\eta1} \,|\, i_\eta; l_{\eta1}) \cdot \mathsf{P}_{\theta_{\text{Vit(G)}}}(x_{\eta1} \,|\, lci_{\eta1}) \\ &= \mathbf{M}(lc_\eta \to lc_{\eta0}\ lc_{\eta1}) \\ &\quad \cdot \mathrm{d}(\mathbf{L}(l_{\eta0}) \cdot \mathbf{e}_{\eta0}) \cdot \mathrm{d}(\mathbf{L}(l_{\eta1}) \cdot \mathbf{e}_{\eta1}) \cdot \mathbf{1} \end{aligned} \tag{10}$$

The syntactic–semantic grammar rule factorization is first applied; the result is rearranged to show how it is equivalent to a chain of vector and matrix products.

The vector alone does not include the syntactic information necessary for the parser. So we need to define a Viterbi probability *over* this vector and relevant syntactic variables:

$$\mathsf{P}_{\theta_{\text{Vit(G)}}}(x_\eta \,|\, lc\mathbf{e}_\eta) \overset{\text{def}}{=} [\![\ \mathbf{e}_\eta = \arg\max_{lc\mathbf{e}_\iota} \mathbf{a}_\iota^T \mathbf{e}_\iota\ ]\!] \tag{11}$$

but are also degenerate semantics (one-word concepts that summarize the information below them). This semantic space definition accomplishes bilexical parsing (Charniak, 1997; Collins, 1997; Eisner and Satta, 1999). A more plausible semantic model is in Section 3.

[3]In some bilexical parsers, the lexical head may be conditioned on phrasal categories — or other local factors such as whether the child is before or after the head. This is done in the present framework by diversifying the set of labeled relations: e.g., $\mathbf{L}(l_{\text{MOD,S,NP}})$.

where $[\![\cdot]\!]$ is an indicator function such that $[\![\phi]\!]=1$ if ϕ is true, 0 otherwise. The deterministic probability means that we will construct the vector according to Eqn. 10 and pick the best children with probability 1. Assuming the rule expansion in the example is the best choice, we will have:

$$\mathsf{P}_{\theta_{\text{Vit(G)}}}(x_0 \,|\, lc\mathbf{e}_0) = [\![\mathbf{e}_0 = \begin{array}{c} \\ \mathrm{h_p} \\ \mathrm{h_t} \\ \mathrm{h_u} \end{array}\!\!\begin{array}{c} \textit{truth} \\ \left[\begin{array}{c} 0 \\ 0 \\ 0.036 \end{array}\right] \end{array}]\!] \qquad (11')$$

In vectorizing the notation, a minor difference has been introduced. Eqn. 4 defined $\mathsf{P}_{\theta_{\text{Vit(G)}}}(x_\eta \,|\, lci_\eta)$ as the maximum over child expansions. However, careful scrutiny of the vectorized version in Eqns. 10 and 11 reveals that the maximum includes π_{G} and occurs outside the per-element definition of $\theta_{\text{Vit(G)}}$.

This approximation allows us to deal with semantic vectors as the canonical form, and is evaluated in Section 4.1. Then, determining the root constituent of the Viterbi tree is the same process as choosing any other Viterbi constituent, maximizing over the posterior.

$$\mathsf{P}(x\; lc_\epsilon) = \mathbf{a}_\epsilon^T \cdot \mathbf{e}_\epsilon \cdot \mathsf{P}_{\pi_{\text{G}\epsilon}}(lc\mathbf{a}_\epsilon^T) \cdot \mathsf{P}_{\theta_{\text{Vit(G)}}}(x \,|\, lc\mathbf{e}_\epsilon) \qquad (12)$$

The first two terms give the encapsulated probabilities, and the last two terms deterministically ensure that the tree τ_ϵ is indeed the one generated. As before, the most likely tree $\hat{\tau}_\epsilon$ is the tree that maximizes this probability, and can be constructed recursively from the best child trees.

Eqns. 6–12 complete the linear algebraic definition of structured vectorial semantics. Notably, $\hat{\tau}_\epsilon$ has an associated vector with a distinct likelihood for each i in $|E|$ which may be construed as the composed vectorial semantic information for the whole parsed sentence. Similar vectors can be obtained anywhere on the parse chart.

3 SVS with Relational Clusters

There are many semantic spaces to choose from to instantiate a structured vectorial semantics (SVS). Headword-lexicalization SVS, as in the example above, uses lexical items as indices ($i \equiv h$) in all of the probability models necessary for SVS: θ_{M}, θ_{L}, $\theta_{\text{P-Vit(G)}}$, $\pi_{\text{G}\epsilon}$, and π_{G}. Based on these relative frequency-estimated models, this section will redefine the SVS equations to deal with relational clustering of headwords.

Typical vector-space models define co-occurrences on documents, but here co-occurrences will be defined such that the approach extends headword-lexicalization SVS. Thus, we will use EM to learn clusters from parent–child headword pairs in the context of the child's (head or modifier) relation, l.

The result will resemble automatic state-splitting techniques, like those described in Matsuzaki et al. (2005). Like those approaches, the resulting 'latent annotations' (i.e., concept vectors) will bear both syntactic and semantic information, but the SVS representation is unique in being able to quantitatively capture a distribution over cohesive concepts at the end of parsing.

The parameters of three probability models will be estimated in EM:

1. $\hat{\mathsf{P}}_{\pi_{\text{E}}}(i_\eta)$, the prior probability of each semantic cluster
2. $\hat{\mathsf{P}}_{\theta_{\text{L}}}(i_{\eta\iota} \,|\, i_\eta; l_{\eta\iota})$, the probability that a parent concept i_η will produce a child concept $i_{\eta\iota}$ if the associated relation is $l_{\eta\iota}$
3. $\hat{\mathsf{P}}_{\theta_{\text{H}}}(h_{\eta\iota} \,|\, i_{\eta\iota})$, a probabilistic mapping from clusters to headwords

In sum, there are $|E|+|E|^2|L|+|E||H|$ parameters to estimate. These are randomly initialized for the Expectation–Maximization (EM) algorithm.

E-step:

$$\hat{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota}) = \cdot\hat{\mathsf{P}}_{\theta_{\text{H}}}(h_\eta \,|\, i_\eta) \cdot \hat{\mathsf{P}}_{\theta_{\text{H}}}(h_{\eta\iota} \,|\, i_{\eta\iota}) \cdot \hat{\mathsf{P}}_{\theta_{\text{L}}}(i_{\eta\iota} \,|\, i_\eta; l_{\eta\iota}) \cdot \tilde{\mathsf{P}}(l_{\eta\iota}) \cdot \hat{\mathsf{P}}_{\pi_{\text{E}}}(i_\eta) \qquad (13)$$

$$\tilde{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota}) = \frac{\hat{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota})}{\sum_{h_\eta, hl_{\eta\iota}} \hat{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota})} \cdot \tilde{\mathsf{P}}(h_\eta, h_{\eta\iota} l_{\eta\iota}) \qquad (14)$$

M-step:

$$\hat{\mathsf{P}}_{\pi_{\text{E}}}(i_\eta) \leftarrow \sum_{h_\eta, ihl_{\eta\iota}} \tilde{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota}) \tag{15}$$

$$\hat{\mathsf{P}}_{\theta_{\text{H}}}(h_\eta \,|\, i_\eta) \leftarrow \frac{\sum_{ihl_{\eta\iota}} \tilde{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota})}{\sum_{h_\eta, ihl_{\eta\iota}} \tilde{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota})} \tag{16}$$

$$\hat{\mathsf{P}}_{\theta_{\text{L}}}(i_{\eta\iota} | \, i_\eta; l_{\eta\iota}) \leftarrow \frac{\sum_{h_\eta, h_{\eta\iota}} \tilde{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota})}{\sum_{h_\eta, ih_{\eta\iota}} \tilde{\mathsf{P}}(ih_\eta, ih_{\eta\iota}\, l_{\eta\iota})} \tag{17}$$

Using the estimated models from EM, we can redefine the parser models θ_{M}, θ_{L}, $\theta_{\text{P-Vit(G)}}$, π_{G}, and $\pi_{\text{G}\epsilon}$. To do so, intermediate distributions $\mathsf{P}(i_\eta \,|\, h_\eta)$ and $\mathsf{P}(h_\eta \,|\, lci_\eta)$ are calculated from estimated models $\hat{\mathsf{P}}(\cdot)$ and frequency-counted models $\tilde{\mathsf{P}}(\cdot)$ using Bayes' rule and marginalization.

First, the relationally-clustered version of syntactic child generation θ_{M} is defined in terms of the headword-lexicalization version of θ_{M}.

$$\hat{\mathsf{P}}_{\theta_{\text{M}}}(lci_\eta \rightarrow lc_{\eta 0} lc_{\eta 1}) = \sum_{h_\eta} \tilde{\mathsf{P}}_{\theta_{\text{M}}}(lch_\eta \rightarrow lc_{\eta 0} lc_{\eta 1}) \cdot \mathsf{P}(h_\eta \,|\, lci_\eta) \tag{18}$$

The clustered model of relations θ_{L} is already trained by the EM algorithm, $\hat{\mathsf{P}}_{\theta_{\text{L}}}(i_{\eta\iota} | \, i_\eta; l_{\eta\iota})$. We might expect it to be rewritten in terms of the lexicalized version of the same model, but that 'rewriting' was already done in the EM algorithm. Clustering is intentionally done on only the semantic portion of the model.

Viterbi likelihoods at preterminal nodes are calculated from the lexicalized version, similar to θ_{M}.

$$\hat{\mathsf{P}}_{\theta_{\text{P-Vit(G)}}}(x_\eta \,|\, lci_\eta) = \sum_{h_\eta} \tilde{\mathsf{P}}_{\theta_{\text{P-Vit(G)}}}(x_\eta \,|\, lch_\eta) \cdot \mathsf{P}(h_\eta \,|\, lci_\eta) \tag{19}$$

Finally, prior probabilities are also derived from lexicalized versions.

$$\hat{\mathsf{P}}_{\pi_{\text{G}\epsilon}}(lci_\epsilon) = \sum_{h_\epsilon} \mathsf{P}(i_\epsilon \,|\, h_\epsilon) \cdot \tilde{\mathsf{P}}_{\pi_{\text{G}\epsilon}}(lch_\epsilon) \tag{20}$$

$$\hat{\mathsf{P}}_{\pi_{\text{G}}}(lci_\eta) = \sum_{h_\eta} \mathsf{P}(i_\eta \,|\, h_\eta) \cdot \tilde{\mathsf{P}}_{\pi_{\text{G}}}(lch_\eta) \tag{21}$$

Sentence probabilities or Viterbi trees can then be found as in Eqn. 5.

Cluster i_1		**Cluster** i_4		**Cluster** i_6		**Cluster** i_9	
unk	0.984	to	0.140	and	0.075	of	0.115
years	0.005	for	0.061	$	0.032	said	0.066
stock	0.003	at	0.056	it	0.020	million	0.057
exchange	0.003	unk	0.042	mr.	0.019	was	0.041
industry	0.002	with	0.031	but	0.016	billion	0.020
service	0.001	as	0.024	an	0.015	company	0.015
tax	0.001	on	0.019	will	0.014	is	0.014
research	0.0006	than	0.019	or	0.011	rose	0.012

Table 1: Sample clusters for the 5,000-headword, 10-concept model, with probabilities from θ_{H}.

It should be clear that the clustered concepts in this parser can be viewed as a generalization of lexicalization. The key is that for the same domain of relations L, clusters share information whereas headwords do not.

For some number of headwords, EM finds a defined (smaller) number of concept clusters. By grouping similar words together, the clusters encapsulate latent semantic information in appropriate relational context l. One might hypothesize that latent information would contain information about word class, semantic role, or some other generalization of lexical items.

4 Evaluation

Sections 02–21 of the Wall Street Journal (WSJ) corpus were used as training data; Section 23 was used for standard parse evaluations. Trees were binarized; then, 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. A set number of headwords (e.g., $h_1, \ldots, h_{50}$) were stored, and the rest were assigned a constant, 'unknown' headword category.

From counts on the binary rules of these annotated trees, the θ_{M}, θ_{L}, and π_{G} probabilities for headword lexicalization 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 they were also inputs to the EM algorithm

WSJ Sec. 23 no punct	LR	LP	F
no vec, 00hw	75.54	76.11	**75.83**
vectorized, 00hw	75.54	76.15	**75.84**
no vec, 10hw	75.46	76.26	**75.86**
vectorized, 10hw	75.45	76.26	**75.85**
no vec, 50hw	76.15	76.77	**76.46**
vectorized, 50hw	76.20	76.85	**76.52**

Table 2: Negligible effect of vectorization for lexicalized models.

for relationally-clustered SVS, which used $|E| = 10$ clusters, seeded random initialization, and a maximum of 100 iterations.

Table 1 shows the top headwords from 4 example clusters for the largest evaluated headword model. The four concepts shown can be seen to have internal similarities and intuitive interpretations, both semantic and syntactic in nature. For example, i_9 seems to prefer words concerning fiscal performance. These clusters may seem less interpretable than latent annotations or POS-splits (Petrov et al., 2006), but it should be again stressed that these concepts remain constant over all syntactic categories.

Note also that the probabilities $\hat{\mathsf{P}}_{\theta_{\mathrm{H}}}(h \,|\, i)$ over different headwords are quite uneven. This is because clusters scope out different shapes and sizes in the semantic space.

Several plain CKY parsers were implemented, each operating with or without vectors (i.e., the factored-model matrix multiplications), and with or without the EM-learned relational clusters. Evaluations were run on 2.83 GHz Intel Core™ 2 Quad processors.

4.1 Effect of Vectorization on Accuracy

Recall that an approximation had to be made in Eqn. 11 in order to allow for Viterbi trees to be calculated on whole vectors, rather than on individual elements of each vector. Table 2 shows that vectorization with approximation produces negligible parsing accuracy differences.

Thus, it is possible for a CKY parser to probabilistically decide between whole vectors, rather than separately considering each concept's best syntactic rules. Since the process of vectorization itself incurs no penalty, different vectorial semantic spaces can be freely applied to the parser.

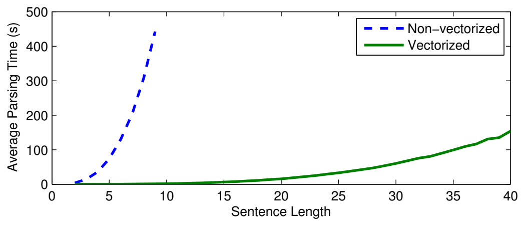


Figure 2: Speed of relationally-clustered SVS parsers, with and without vectorization.

4.2 Effect of Vectorization on Speed

A CKY parser, as used for this implementation, has worst-case $\mathcal{O}(n^3 \cdot |L|^3 |C|^3 |E|^3)$ runtime. While this may be acceptable in typical lexicalized models, the factorization from Section 2.2 additionally requires up to $|E|^2$ multiplications for each syntactic rule. With dense relational distributions, clusters incur a larger cost than the comparatively sparse information found in bilexical dependencies.

Of course, both models may be sped up considerably by using coarse-to-fine parsing, pruning, and other methods. But for vectorial models, contiguous storage of information intuitively leads to more efficient multiplication of relevant probabilities. Thus, improvements in parsing speed for dense, relationally-clustered semantics may be an implementational windfall from vectorization.

The plot in Figure 2 compares simple CKY implementations of a relationally-clustered SVS model with and without vectorization on sentences (ordered by length) on WSJ Section 23 without punctuation. The un-vectorized, relationally-clustered model was cut off because it was impractically slow on longer sentences. Regression analysis shows that these are indeed cubic; the constant in front of $\mathcal{O}(n^3)$ is 0.66505 without vectorization, and 0.00267 with vectorization. In other words, the cost of accessing the grammatical constituents and performing the factored multiplications was greatly reduced by vectorization.

WSJ Sec. 23 no punct ≤ 40	LR	LP	F
syntax only:	76.25	76.83	76.54
baseline 10hw:	76.24	77.04	76.64
baseline 50hw:	76.84	77.48	77.16
baseline 100hw:	77.30	77.71	77.57
50hw → 10 clust:	75.29	75.75	75.52
100hw → 10 clust:	76.69	77.20	76.95
500hw → 10 clust:	78.13	78.19	78.16
1000hw → 10 clust:	79.54	79.78	79.66
5000hw → 10 clust:	**80.89**	**80.83**	**80.86**

WSJ Sec. 22 punct ≤ 40	LR	LP	F
syntax only :	80.3	79.9	80.1
1000hw → 10 clust:	**84.36**	**84.06**	**84.21**

Table 3: Recall, precision, and F-score for unsmoothed lexicalized CKY parsers (top) versus parsers with 10 semantic clusters (bottom).

4.3 Effect of SVS Models

The most demonstrative evaluations for this paper compare head-to-head parsing accuracy for syntax-only parsing, headword-lexicalized SVS, and relationally-clustered SVS. Similar to evaluations on latent-annotation parsing, the baseline syntax case is unadorned, with only a head-preserving (Magerman, 1995) binarization.

In this way, the contribution of the psycholinguistically plausible vector space will not be confused with other methods that implicitly model some semantics (e.g., POS splitting). Table 3 shows precision, recall, and F-score for each of the models.

First, consider whether SVS faithfully performs lexicalized parsing. With 10 headwords in headword-lexicalization SVS, there is an insignificant increase in parser accuracy, but at 50 and 100 headwords the gains in lexicalization are consistent with existing literature (Gildea, 2001). Thus, the syntactic–semantic factorization in SVS does not invalidate the expected gains of lexicalization.

Next, semantics with 50 headwords clustered into 10 concepts decreases performance from even an unlexicalized baseline. This is likely due to degenerate cluster formation, since the occurrence of the "unk" in 50-headword trees dominates all other headwords. The effect is still somewhat present with 100 headwords.

However, a randomly-initialized 5,000-headword, 10-cluster model vastly outperforms a 10-concept lexicalized model, even though the two have theoretically equivalent state spaces. Increasing the headwords in a lexicalized model leads to a plateau in efficacy, so no comparison to a 5,000-headword lexicalized model was done.

Moreover, this 10-cluster model outperforms an unlexicalized baseline (without punctuation) by 4.22 percentage points; performance is preserved over a higher baseline (with punctuation) using a similar model to obtain a 4.11 absolute point gain.

The relationally-clustered models bear much more rich and useful information for parsing than do headwords. Of course, some of these gains are from the implied syntactic state-splitting, since SVS blurs the lines between syntax and semantics. But composing vectors that have cohesive semantic representation may be hypothesized to model selectional preferences and other semantics-level processes that are not explicitly modeled in other approaches.

5 Conclusion and Future Work

This paper has introduced a structured vectorial semantic modeling framework that fully interleaves a coherent vectorial semantic component with syntactic parsing. The language model was motivated with probabilities in standard PCFG parsing, but these probabilities were then encapsulated in vectors and matrices. Vectorization proved to have no deleterious effect on parsing accuracy; rather, it improved parsing speed.

Two instantiations were explored and evaluated: headword-lexicalization SVS and relational-clustering SVS. It was found that relationally-clustered SVS vastly outperformed the simpler lexicalized model, partially accomplishing automatic state-splits while still maintaining a coherent semantic representation over all constituents.

Finally, it is hoped that this plausible, efficient framework will enable new generations of cognitive models that allow predictions from syntactic and (latent) semantic factors to naturally interact in comprehension.

References

Banerjee, Arindam, Sugato Basu, and Srujana Merugu. 2007. Multi-way clustering on relation graphs. In *Proceedings of the SIAM International Conference on Data Mining (SDM-2007)*, April.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Boyd-Graber, Jordan and David M. Blei. 2008. Syntactic topic models. In *Neural Information Processing Systems*.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Fourteenth National Conference on Artificial Intelligence*, Providence, Rhode Island.

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

Deerwester, Scott, Susan Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Eisner, Jason and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 457–464. Association for Computational Linguistics.

Erk, Katrin and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP 2008*.

Gildea, Daniel. 2001. Corpus variation and parser performance. In *In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP '00)*, Pittsburgh, PA.

Griffiths, Thomas L., Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. *Advances in neural information processing systems*, 17:537–544.

Hofmann, Thomas. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

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

Kintsch, Walter. 2001. Predication. *Cognitive Science*, 25(2):173–202.

Landauer, T.K. and S.T. Dumais. 1997. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104:211–240.

Lund, K. and C. Burgess. 1996. Producing High-Dimensional Semantic Spaces From Lexical Co-Occurrence. *Behavior Research Methods Instruments and Computers*, 28:203–208.

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

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics.

Mitchell, Jeff and Mirella Lapata. 2009. Language Models Based on Semantic Composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 430–439.

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

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan 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)*.

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