Draft Paper

Learning Syntactic Categories Using Paradigmatic Representations of Word Context

1. Introduction

Grammar rules apply not to individual words (e.g. dog, eat) but to syntactic categories of words (e.g. noun, verb). Thus constructing syntactic categories (also known as lexical or part-of-speech categories) is one of the fundamental problems in language acquisition.

Syntactic categories represent groups of words that can be substituted for one another without altering the grammaticality of a sentence. Linguists identify syntactic categories based on semantic, syntactic, and morphological properties of words. There is also evidence that children use prosodic and phonological features to bootstrap syntactic category acquisition (**?**). However there is as yet no satisfactory computational model that can match human performance. Thus identifying the best set of features and best learning algorithms for syntactic category acquisition is still an open problem.

Relationships between linguistic units can be classified into two types: syntagmatic (concerning positioning), and paradigmatic (concerning substitution). Syntagmatic relations determine which units can combine to create larger groups and paradigmatic relations determine which units can be substituted for one another. Figure 1 illustrates the paradigmatic vs syntagmatic axes for words in a simple sentence and their possible substitutes.

In this study, we represent the paradigmatic axis directly by building *substitute vectors* for each word position in the text. The dimensions of a substitute vector represent words in the

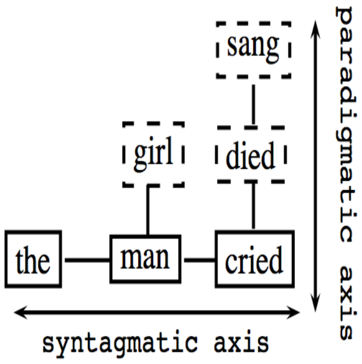

Figure 1
Syntagmatic vs. paradigmatic axes for words in a simple sentence (**?**).

vocabulary, and the magnitudes represent the probability of occurrence in the given position. Note that the substitute vector for a word position (e.g. the second word in Fig. 1) is a function of the context only (i.e. "the ___ cried"), and does not depend on the word that does actually appear there (i.e. "man"). Thus substitute vectors represent *individual word contexts*, not word types. We refer to the use of features based on substitute vectors as *paradigmatic representations of word context*.

Our preliminary experiments indicated that using context information alone without the identity or the features of the target word (e.g. using dimensionality reduction and clustering on substitute vectors) has limited success and modeling the co-occurrence of word and context types is essential for inducing syntactic categories. In the models presented in this paper, we combine paradigmatic representations of word context with features of co-occurring words within the co-occurrence data embedding (CODE) framework (**?**; **?**). The resulting embeddings for word types are split into 45 clusters using k-means and the clusters are compared to the 45 gold tags in the 1M word Penn Treebank Wall Street Journal corpus (**?**). We obtain many-to-one accuracies up to .7680 using only distributional information (the identity of the word and a representation of its context) and .8023 using morphological and orthographic features of words improving the state-of-the-art in unsupervised part-of-speech tagging performance.

The high probability substitutes reflect both semantic and syntactic properties of the context as seen in the example below (the numbers in parentheses give substitute probabilities):

"Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29."
the: its (.9011), the (.0981), a (.0006), . . .
board: board (.4288), company (.2584), firm (.2024), bank (.0731), . . .

Top substitutes for the word "the" consist of words that can act as determiners. Top substitutes for "board" are not only nouns, but specifically nouns compatible with the semantic context.

This example illustrates two concerns inherent in all distributional methods: (i) words that are generally substitutable like "the" and "its" are placed in separate categories (DT and PRP$) by the gold standard, (ii) words that are generally not substitutable like "do" and "put" are placed in the same category (VB). Freudenthal et al. (**?**) point out that categories with unsubstitutable words fail the standard linguistic definition of a syntactic category and children do not seem to make errors of substituting such words in utterances (e.g. *"What do you want?"* vs. **"What put you want?"*). Whether gold standard part-of-speech tags or distributional categories are better suited to applications like parsing or machine translation can be best decided using extrinsic evaluation. However in this study we follow previous work and evaluate our results by comparing them to gold standard part-of-speech tags.

Section 2 gives a detailed review of related work. Section 3 describes the dataset and the construction of the substitute vectors. Section 8 describes co-occurrence data embedding, the learning algorithm used in our experiments. Section 9 describes our experiments and compares our results with previous work. Section 11 gives a brief error analysis and Section **??** summarizes our contributions. All the data and the code to replicate the results given in this paper is available from the authors' website at `http://goo.gl/RoqEh`.

2. Related Work

There are several good reviews of algorithms for unsupervised part-of-speech induction (**?**; **?**) and models of syntactic category acquisition (**?**).

This work is to be distinguished from supervised part-of-speech disambiguation systems, which use labeled training data (**?**), unsupervised disambiguation systems, which use a dictionary of possible tags for each word (**?**), or prototype driven systems which use a small set of prototypes for each class (**?**). The problem of induction is important for studying under-resourced languages that lack labeled corpora and high quality dictionaries. It is also essential in modeling child language acquisition because every child manages to induce syntactic categories without access to labeled sentences, labeled prototypes, or dictionary constraints.

Models of unsupervised part-of-speech induction fall into two broad groups based on the information they utilize. Distributional models only use word types and their context statistics. Word-feature models incorporate additional morphological and orthographic features.

2.1 Distributional models

Distributional models can be further categorized into three subgroups based on the learning algorithm. The first subgroup represents each word type with its context vector and clusters these vectors accordingly (**?**). Work in modeling child syntactic category acquisition has generally followed this clustering approach (**?**; **?**). The second subgroup consists of probabilistic models based on the Hidden Markov Model (HMM) framework (**?**). A third group of algorithms constructs a low dimensional representation of the data that represents the empirical co-occurrence statistics of word types (**?**), which is covered in more detail in Section 8.

Clustering:. Clustering based methods represent context using neighboring words, typically a single word on the left and a single word on the right called a "frame" (e.g., **the** *dog* **is***;* **the** *cat* **is**). They cluster word types rather than word tokens based on the frames they occupy thus employing one-tag-per-word assumption from the beginning (with the exception of some methods in (**?**)). They may suffer from data sparsity caused by infrequent words and infrequent contexts. The solutions suggested either restrict the set of words and set of contexts to be clustered to the most frequently observed, or use dimensionality reduction. Redington et al. (**?**) define context similarity based on the number of common frames bypassing the data sparsity problem but achieve mediocre results. Mintz (**?**) only uses the most frequent 45 frames and Biemann (**?**) clusters the most frequent 10,000 words using contexts formed from the most frequent 150-200 words. Schütze (**?**) and Lamar et al. (**?**) employ SVD to enhance similarity between less frequently observed words and contexts. Lamar et al. (**?**) represent each context by the currently assigned left and right tag (which eliminates data sparsity) and cluster word types using a soft k-means style iterative algorithm. They report the best clustering result to date of .708 many-to-one accuracy on a 45-tag 1M word corpus.

HMMs:. The prototypical bitag HMM model maximizes the likelihood of the corpus $w_1 \dots w_n$ expressed as $P(w_1|c_1) \prod_{i=2}^{n} P(w_i|c_i)P(c_i|c_{i-1})$ where w_i are the word tokens and c_i are their (hidden) tags. One problem with such a model is its tendency to distribute probabilities equally and the resulting inability to model highly skewed word-tag distributions observed in hand-labeled data (**?**). To favor sparse word-tag distributions one can enforce a strict one-tag-per-word solution (**?**; **?**), use sparse priors in a Bayesian setting (**?**; **?**), or use posterior regularization (**?**). Each of these techniques provide significant improvements over the standard HMM model: for example Gao and Johnson (**?**) show that sparse priors can gain from 4% (.62 to .66 with a 1M word corpus) in cross-validated many-to-one accuracy. However Christodoulopoulos et al. (**?**) show that the older one-tag-per-word models such as (**?**) outperform the more sophisticated sparse prior and posterior regularization methods both in speed and accuracy (the Brown model gets .68 many-to-one accuracy with a 1M word corpus). Given that close to 95% of the word occurrences in human labeled data are tagged with their most frequent part of speech (**?**), this is probably not surprising; one-tag-per-word is a fairly good first approximation for induction.

2.2 Word-feature models

One problem with the algorithms in the previous section is the poverty of their input features. Of the syntactic, semantic, and morphological information linguists claim underlie syntactic categories, context vectors or bitag HMMs only represent limited syntactic information in their input. Experiments incorporating morphological and orthographic features into HMM based models demonstrate significant improvements. (**?**; **?**; **?**) incorporate similar orthographic features and report improvements of 3, 7, and 10% respectively over the baseline Brown model. Christodoulopoulos et al. (**?**) use prototype based features as described in (**?**) with automatically induced prototypes and report an 8% improvement over the baseline Brown model. Christodoulopoulos et al. (**?**) define a type-based Bayesian multinomial mixture model in which each word instance is generated from the corresponding word type mixture component and word contexts are represented as features. They achieve a .728 MTO score by extending their model with additional morphological and alignment features gathered from parallel corpora. To our knowledge, nobody has yet tried to incorporate phonological or prosodic features in a computational model for syntactic category acquisition.

2.3 Paradigmatic representations

Sahlgren (**?**) gives a detailed analysis of paradigmatic and syntagmatic relations in the context of word-space models used to represent word meaning. Sahlgren's paradigmatic model represents word types using co-occurrence counts of their frequent neighbors, in contrast to his syntagmatic model that represents word types using counts of contexts (documents, sentences) they occur in. Our substitute vectors do not represent word types at all, but *contexts of word tokens* using probabilities of likely substitutes. Sahlgren finds that in word-spaces built by frequent neighbor vectors, more nearest neighbors share the same part-of-speech compared to word-spaces built by context vectors. We find that representing the paradigmatic axis more directly using substitute vectors rather than frequent neighbors improve part-of-speech induction.

Our paradigmatic representation is also related to the second order co-occurrences used in (**?**). Schütze concatenates the left and right context vectors for the target word type with the left context vector of the right neighbor and the right context vector of the left neighbor. The vectors from the neighbors include potential substitutes. Our method improves on his foundation by using a 4-gram language model rather than bigram statistics, using the whole 78,498 word vocabulary rather than the most frequent 250 words. More importantly, rather than simply concatenating vectors that represent the target word with vectors that represent the context we use S-CODE to model their co-occurrence statistics.

2.4 Evaluation

We report many-to-one and V-measure scores for our experiments as suggested in (**?**). The many-to-one (MTO) evaluation maps each cluster to its most frequent gold tag and reports the percentage of correctly tagged instances. The MTO score naturally gets higher with increasing number of clusters but it is an intuitive metric when comparing results with the same number of clusters. The V-measure (VM) (**?**) is an information theoretic metric that reports the harmonic mean of homogeneity (each cluster should contain only instances of a single class) and completeness (all instances of a class should be members of the same cluster). In Section 11 we argue that homogeneity is perhaps more important in part-of-speech induction and suggest MTO with a fixed number of clusters as a more intuitive metric.

3. Substitute Vectors

In this study, we predict the part of speech of a word in a given context based on its substitute vector. The dimensions of the substitute vector represent words in the vocabulary, and the entries in the substitute vector represent the probability of those words being used in the given context. Note that the substitute vector is a function of the context only and is indifferent to the target word. This section details the choice of the data set, the vocabulary and the estimation of substitute vector probabilities.

The Wall Street Journal Section of the Penn Treebank (**?**) was used as the test corpus (1,173,766 tokens, 49,206 types). The treebank uses 45 part-of-speech tags which is the set we used as the gold standard for comparison in our experiments. To compute substitute probabilities we trained a language model using approximately 126 million tokens of Wall Street Journal data (1987-1994) extracted from CSR-III Text (**?**) (we excluded the test corpus). We used SRILM (**?**) to build a 4-gram language model with Kneser-Ney discounting. Words that were observed less than 20 times in the language model training data were replaced by UNK tags, which gave us a vocabulary size of 78,498. The perplexity of the 4-gram language model on the test corpus is 96.

It is best to use both left and right context when estimating the probabilities for potential lexical substitutes. For example, in *"He lived in San Francisco suburbs."*, the token *San* would be difficult to guess from the left context but it is almost certain looking at the right context. We define c_w as the $2n-1$ word window centered around the target word position: $w_{-n+1} \dots w_0 \dots w_{n-1}$ ($n = 4$ is the n-gram order). The probability of a substitute word w in a given context c_w can be estimated as:

$$\begin{aligned} P(w_0 = w|c_w) &\propto P(w_{-n+1} \dots w_0 \dots w_{n-1}) && (1)\\ &= P(w_{-n+1})P(w_{-n+2}|w_{-n+1}) \\ &\quad \dots P(w_{n-1}|w_{-n+1}^{n-2}) && (2)\\ &\approx P(w_0|w_{-n+1}^{-1})P(w_1|w_{-n+2}^{0}) \\ &\quad \dots P(w_{n-1}|w_0^{n-2}) && (3) \end{aligned}$$

where w_i^j represents the sequence of words $w_i w_{i+1} \dots w_j$. In Equation 1, $P(w|c_w)$ is proportional to $P(w_{-n+1} \dots w_0 \dots w_{n+1})$ because the words of the context are fixed. Terms without w_0 are identical for each substitute in Equation 2 therefore they have been dropped in Equation 3. Finally, because of the Markov property of n-gram language model, only the closest $n-1$ words are used in the experiments.

Near the sentence boundaries the appropriate terms were truncated in Equation 3. Specifically, at the beginning of the sentence shorter n-gram contexts were used and at the end of the sentence terms beyond the end-of-sentence token were dropped.

For computational efficiency only the top 100 substitutes and their unnormalized probabilities were computed for each of the 1,173,766 positions in the test set[1]. The probability vectors for each position were normalized to add up to 1.0 giving us the final substitute vectors used in the rest of this study.

1 The substitutes with unnormalized log probabilities can be downloaded from `http://goo.gl/jzKH0`. For a description of the FASTSUBS algorithm used to generate the substitutes please see `http://arxiv.org/abs/1205.5407v1`. FASTSUBS accomplishes this task in about 5 hours, a naive algorithm that looks at the whole vocabulary would take more than 6 days on a typical 2012 workstation.

4. Distance Metric

We represent each context with a sparse high dimensional probability vector called the substitute vector as described in the previous section. In this section we compare various distance metrics in this high dimensional space with the goal of discovering one that will judge vectors that belong to the same syntactic category similar and vectors that belong to different syntactic categories distant. The distance metrics we have considered are listed in Table 1.

Cosine($\mathbf{p}, \mathbf{q}$)	=	$< \mathbf{p}, \mathbf{q} > /(\|\mathbf{p}\|_2 \|\mathbf{q}\|_2)$
Euclid($\mathbf{p}, \mathbf{q}$)	=	$\|\mathbf{p} - \mathbf{q}\|_2$
Manhattan($\mathbf{p}, \mathbf{q}$)	=	$\|\mathbf{p} - \mathbf{q}\|_1$
Maximum($\mathbf{p}, \mathbf{q}$)	=	$\|\mathbf{p} - \mathbf{q}\|_\infty$
KL2($\mathbf{p}, \mathbf{q}$)	=	$\sum_i p_i ln(p_i/q_i) + q_i ln(q_i/p_i)$
JS($\mathbf{p}, \mathbf{q}$)	=	$\sum_i p_i ln(p_i/m_i) + q_i ln(q_i/m_i)$
		where $m_i = (p_i + q_i)/2$

Table 1
Similarity metrics. JS is the Jensen-Shannon divergence and KL2 is a symmetric implementation of Kullback-Leibler divergence.

To judge the merit of each distance metric we obtained supervised baseline scores using leave-one-out cross validation and the weighted k-nearest-neighbor algorithm[2] on the gold tags of the 1M word WSJ corpus. The results are listed in Table 2 sorted by score.

Metric	Accuracy(%)
KL2	0.6889
Manhattan	0.6865
Jensen	0.6801
Cosine	0.6706
Maximum	0.6663
Euclid	0.6255
lg2-Maximum	0.5361
lg2-Cosine	0.4847
lg2-Euclid	0.4038
lg2-Manhattan	0.3729

Table 2
Supervised baseline scores with different distance metrics. Log-metric indicates that metric applied to the log of the probability vectors.

The entries with the log- prefix indicate a metric applied to the log of the probability vectors. Distance metrics on log probability vectors performed poorly compared to their regular counterparts indicating differences in low probability words are relatively unimportant and high probability substitutes determine syntactic category. The surprisingly good result achieved by the simple Maximum metric (which identifies the dimension with the largest difference between two vectors) also support this conclusion. The maximum score of .73% can be taken as a rough upper bound for an unsupervised learner using this space on the 45-tag 1M word WSJ corpus because .27% of the instances are assigned to the wrong part of speech by the majority of their

2 Neighbors were weighted using 1/distance, $k = 30$ was chosen empirically.

closest neighbors. We will discuss ways to push this upper bound higher by including word type information together with other features in Section **??** and Section **??**, respectively.

5. Dimensionality Reduction

Even with sparsity using high dimensional vectors is problematic with many learning algorithms because of computational cost and the curse of dimensionality. In this section we investigate if there is a low dimensional representation of the substitute vectors which still preserve the neighborhood information necessary to learn syntactic categories. We first briefly describe then report experimental results on principal components analysis (PCA), Isomap (**?**), locally linear embedding (LLE) (**?**), and Laplacian eigenmaps (**?**).

Each dimensionality reduction algorithm tries to preserve certain aspects of the original vectors. PCA is a linear method that minimizes reconstruction error. Isomap tries to preserve distances as measured along a low dimensional submanifold assuming the input vectors were sampled from the neighborhood of such a manifold. LLE most faithfully preserves the local linear structure of nearby input vectors. Laplacian eigenmaps most faithfully preserve proximity relations, mapping nearby inputs to nearby outputs.

We wanted to see how accuracy (based on the k-nearest-neighbor supervised baseline as in the previous section) changed based on the number of dimensions for each dimensionality reduction algorithm. Even with sparse representations calculations with matrices of size one million is computationally very hard, however. Hence we built 10 substitute vector sets for 24k chunks of contiguous segments extracted from the Wall Street Journal Section of the Penn Treebak (**?**). We applied each algorithm to each chunk and obtained average accuracy and standard deviation for each number of dimensions. All the parameters were set empirically to values that gave reasonable results: For algorithms that require a distance matrix rather than raw input vectors we used the Jensen-Shannon divergence judged best by the experiments of the previous section. For graph based methods we built neighborhood graphs using 100 nearest neighbors. The low dimensional output vectors were compared using the cosine distance metric for the supervised k-nearest-neighbor algorithm. Figure 2 plots supervised baseline accuracy vs. number of dimensions for each algorithm.

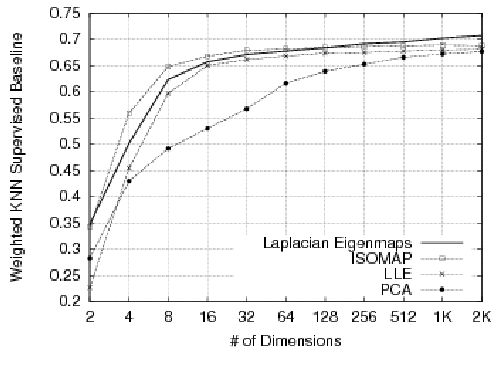

Figure 2
Supervised knn baselines for the four dimensionality reduction algorithms.

The graph based algorithms (Isomap, LLE, and Laplacian eigenmaps) all outperform PCA. They stay within 5% of their peak accuracy with as few as 16 dimensions. In fact Laplacian eigenmaps outperform the baseline with the original 12,672 dimensional vectors (68.95%) when allowed to retain more than about 250 dimensions. Spectral clustering uses the same transformation as the Laplacian eigenmaps algorithm and we compare its performance to other clustering algorithms in the next section.

6. Clustering

We compared three clustering algorithms applied to the original substitute vectors using many-to-one accuracy on the 45-tag 24K word test corpus. Hierarchical agglomerative clustering with complete linkage (HAC) starts with each instance in its own cluster and iteratively combines the two closest groups (measured by their most distant points) at each step (**?**). K-medoids minimizes sum of pairwise distances between each datapoint to the exemplar at the center of its cluster (**?**). Spectral clustering[3] uses the eigenvalues of the graph Laplacian $L = D^{-1/2}WD^{-1/2}$ to reduce the number of dimensions (similar to Laplacian eigenmaps) and uses simple k-means clustering on the resulting representation (**?**). All three algorithms accept the distance matrix based on the KL2 distance (see Section 4) as input.

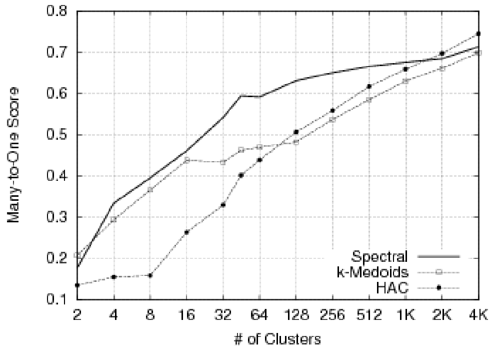

Figure 3
Many-to-one score for three clustering algorithms on the 45-tag 24K word corpus.

Figure 3 plots the many-to-one score versus number of clusters for the three algorithms on the 45-tag 24K word test corpus. The many-to-one score naturally increases as we approach the one cluster per word limit, however we find the evolution of the curves informative. At the high end (more than 2000 clusters) HAC performs best with its conservative clusters, but its performance degrades fast as we reduce the number of clusters because it cannot reverse the accumulating mistakes. At the low end (less than 16 clusters) k-medoids and spectral have similar performance. However for the region of interest (between 16 to 2000 clusters) spectral clustering is clearly superior with % many-to-one accuracy at 45 clusters.

7. WSJ Results

In this section, we expanded our test corpus to the complete Wall Street Journal Section of the Penn Treebank (**?**) (1,173,766 tokens, 49,206 types) and applied spectral clustering with 45 clusters. New language model is trained with the same settings in Section 3, with exception of observation threshold for words. Threshold dropped to 20 and vocabulary size increased to 78,498. The perplexity of the 4-gram language model on the test corpus is 96.

Only the top 100 substitutes and their unnormalized probabilities were computed for each of the 1,173,766 positions in the test set since low probability substitutes are relatively unimportant,

3 We used the implementation in (**?**) with a symmetric sparse affinity matrix of 550 nearest neighbors.

as resulted in Section **??**, and computational efficiency is a serious concern with the data[4]. The probability vectors for each position were normalized to add up to 1.0 giving us the final substitute vectors used in the rest of this study.

Supervised baseline scores recomputed for distance metrics, with absence of KL2, which is undefined in the sparse setting, and log space distances, which previously performed poorly (REFERENCE TO APPENDIX HERE). Score ordering found to be in line with results from Section 4 and KL2's successor Manhattan metric is chosen for further computations.

In this paragraph we talk about results of spectral clustering with 45 clusters. This is placeholder.

8. Co-occurrence Data Embedding

The general strategy we follow for unsupervised syntactic category acquisition is to combine features of the context with the identity and features of the target word. Our preliminary experiments in Section 6 indicated that using the context information alone (e.g. clustering substitute vectors) without the target word identity and features had limited success.[5] It is the co-occurrence of a target word with a particular type of context that best predicts the syntactic category. In this section we review the unsupervised methods we used to model co-occurrence statistics: the Co-occurrence Data Embedding (CODE) method (**?**) and its spherical extension (S-CODE) introduced by (**?**). In Section **??** we extend the S-CODE to handle more than two variables.

Let X and Y be two categorical variables with finite cardinalities $|X|$ and $|Y|$. We observe a set of pairs $\{x_i, y_i\}_{i=1}^n$ drawn IID from the joint distribution of X and Y. The basic idea behind CODE and related methods is to represent (embed) each value of X and each value of Y as points in a common low dimensional Euclidean space $\mathbf{R}^d$ such that values that frequently co-occur lie close to each other. There are several ways to formalize the relationship between the distances and co-occurrence statistics, in this paper we use the following:

$$p(x, y) = \frac{1}{Z}\bar{p}(x)\bar{p}(y)e^{-d^2_{x,y}} \tag{4}$$

where $d^2_{x,y}$ is the squared distance between the embeddings of x and y, $\bar{p}(x)$ and $\bar{p}(y)$ are empirical probabilities, and $Z = \sum_{x,y} \bar{p}(x)\bar{p}(y)e^{-d^2_{x,y}}$ is a normalization term. If we use the notation ϕ_x for the point corresponding to x and ψ_y for the point corresponding to y then

4 The substitutes with unnormalized log probabilities can be downloaded from `http://goo.gl/jzKH0`. For a description of the FASTSUBS algorithm used to generate the substitutes please see `http://arxiv.org/abs/1205.5407v1`. FASTSUBS accomplishes this task in about 5 hours, a naive algorithm that looks at the whole vocabulary would take more than 6 days on a typical 2012 workstation.

5 A 10-nearest-neighbor supervised baseline using cosine distance between substitute vectors gives .7213 accuracy. Clustering substitute vectors using various distance metrics and dimensionality reduction methods give results inferior to this upper bound.

$d^2_{x,y} = \|\phi_x - \psi_y\|^2$. The log-likelihood of a given embedding $\ell(\phi, \psi)$ can be expressed as:

$$\begin{aligned}\ell(\phi,\psi) &= \sum_{x,y} \bar{p}(x,y)\log p(x,y) \qquad (5)\\ &= \sum_{x,y} \bar{p}(x,y)(-\log Z + \log \bar{p}(x)\bar{p}(y) - d^2_{x,y})\\ &= -\log Z + const - \sum_{x,y}\bar{p}(x,y)d^2_{x,y}\end{aligned}$$

The likelihood is not convex in ϕ and ψ. We use gradient ascent to find an approximate solution for a set of ϕ_x, ψ_y that maximize the likelihood. The gradient of the $d^2_{x,y}$ term pulls neighbors closer in proportion to the empirical joint probability:

$$\frac{\partial}{\partial \phi_x}\sum_{x,y} -\bar{p}(x,y)d^2_{x,y} = \sum_y 2\bar{p}(x,y)(\psi_y - \phi_x) \qquad (6)$$

The gradient of the Z term pushes neighbors apart in proportion to the estimated joint probability:

$$\frac{\partial}{\partial \phi_x}(-\log Z) = \sum_y 2p(x,y)(\phi_x - \psi_y) \qquad (7)$$

Thus the net effect is to pull pairs together if their estimated probability is less than the empirical probability and to push them apart otherwise. The gradients with respect to ψ_y are similar.

S-CODE (**?**) additionally restricts all ϕ_x and ψ_y to lie on the unit sphere. With this restriction, Z stays around a fixed value during gradient ascent. This allows S-CODE to substitute an approximate constant $\tilde{Z}$ in gradient calculations for the real Z for computational efficiency. In our experiments, we used S-CODE with its sampling based stochastic gradient ascent algorithm and smoothly decreasing learning rate.

8.1 S-Code with More than Two Variables

S-CODE handles two variables, whereas underlying syntactic categories can be captured by more than two different variables such as contextual, morphologic and ortographic features. S-CODE can be extented to handle more than two variables in a way similar to the multi variable extension of CODE (**?**) except that the unit sphere restriction. The log-likelihood at Equation 5 can be redefined for $n+1$ different categorical variables X, Y_i, ... and Y_n with finite cardinalities $|X|$, $|Y_1|$, ... and $|Y_n|$, respectively, as:

$$\begin{aligned}\ell(\phi,\psi_1,\ldots,\psi_n) &= \sum_{i=1}^{n}\sum_{x,y_i}\bar{p}(x,y_i)\log p(x,y_i) \qquad (8)\\ &= \sum_{i=1}^{n}\sum_{x,y_i}\bar{p}(x,y_i)(-\log Z_i + \log\bar{p}(x)\bar{p}(y_i) - d^2_{x,y_i})\\ &= -\sum_{i=1}^{n}(\log Z_i + const_i - \sum_{x,y_i}\bar{p}(x,y_i)d^2_{x,y_i})\end{aligned}$$

Distributional Models	MTO	VM
(**?**)	.708	-
(**?**)*	.678	.630
(Goldwater et al., 2007)	.632	.562
(**?**)*	.625	.548
(**?**)	.688 (.0016)	-
Bigrams (Sec. 9.1)	.7314 (.0096)	.6558 (.0052)
Partitions (Sec. 9.2)	.7554 (.0055)	.6703 (.0037)
Substitutes (Sec. 9.3)	.7680 (.0038)	.6822 (.0029)

Models with Additional Features	MTO	VM
(**?**)*	.712	.655
(**?**)	.728	.661
(**?**)	.755	-
(**?**)	.761	.688
(**?**)	.775	.697
Substitutes and Features (Sec. 9.4)	.8004 (.0075)	.7160 (.0044)

Table 3
Summary of results in terms of the MTO and VM scores. Standard errors are given in parentheses when available. Starred entries have been reported in the review paper (**?**). Distributional models use only the identity of the target word and its context. The models on the right incorporate orthographic and morphological features.

where ψ_i is the embedding of $y_i \in Y_i$ and Z_i is the normalization term of $p(x, y_i)$. Thus the model is able to jointly learn the embeddings when the pairwise co-occurence statistics, $\bar{p}(x, y_i)$, are available for all i.

One problem with these setting is, not every (x, y_i) pair is observed in the data. For example, the stem word "**car**" doesn't have any morphological feature, thus its morphological feature is represented by a null value, "X". However setting the unobserved features to "X" leads to pulling the words with unobserved features together while pushing the ones with observed features apart. To solve this, during the gradient search we don't perform any pull or push update on embeddings if the value of y_i is set to null. [check the the code rewrite]

9. Experiments

In this section we present experiments that evaluate substitute vectors as representations of word context within the S-CODE framework. Section 9.1 replicates the bigram based S-CODE results from (**?**) as a baseline. The S-CODE algorithm works with discrete inputs. The substitute vectors as described in Section 3 are high dimensional and continuous. We experimented with two approaches to use substitute vectors in a discrete setting. Section 9.2 presents an algorithm that partitions the high dimensional space of substitute vectors into small neighborhoods and uses the partition id as a discrete context representation. Section 9.3 presents an even simpler model which pairs each word with a random substitute. When the left-word – right-word pairs used in the bigram model are replaced with word – partition-id or word – substitute pairs we see significant gains in accuracy. These results support our running hypothesis that paradigmatic features, i.e. potential substitutes of a word, are better determiners of syntactic category compared to left and right neighbors. Section 9.4 explores morphologic and orthographic features as additional sources of information and its results improve the state-of-the-art in the field of unsupervised syntactic category acquisition.

Each experiment was repeated 10 times with different random seeds and the results are reported with standard errors in parentheses or error bars in graphs. Table 3 summarizes all the results reported in this paper and the ones we cite from the literature.

9.1 Bigram model

In (?) adjacent word pairs (bigrams) in the corpus are fed into the S-CODE algorithm as X, Y samples. The algorithm uses stochastic gradient ascent to find the ϕ_x, ψ_y embeddings for left and right words in these bigrams on a single 25-dimensional sphere. At the end each word w in the vocabulary ends up with two points on the sphere, a ϕ_w point representing the behavior of w as the left word of a bigram and a ψ_w point representing it as the right word. The two vectors for w are concatenated to create a 50-dimensional representation at the end. These 50-dimensional vectors are clustered using an instance weighted k-means algorithm and the resulting groups are compared to the correct part-of-speech tags. Maron et al. (?) report many-to-one scores of .6880 (.0016) for 45 clusters and .7150 (.0060) for 50 clusters (on the full PTB45 tag-set). If only ϕ_w vectors are clustered without concatenation we found the performance drops significantly to about .62.

To make a meaningful comparison we re-ran the bigram experiments using our default settings and obtained a many-to-one score of .7314 (.0096) and the V-measure is .6558 (.0052) for 45 clusters. The following default settings were used: (i) each word was kept with its original capitalization, (ii) the learning rate parameters were adjusted to $\varphi_0 = 50$, $\eta_0 = 0.2$ for faster convergence in log likelihood, (iii) the number of s-code iterations were increased from 12 to 50 million, (iv) k-means initialization was improved using (?), and (v) the number of k-means restarts were increased to 128 to improve clustering and reduce variance.

9.2 Random partitions

Instead of using left-word – right-word pairs as inputs to S-CODE we wanted to pair each word with a paradigmatic representation of its context to get a direct comparison of the two context representations. To obtain a discrete representation of the context, the random–partitions algorithm first designates a random subset of substitute vectors as centroids to partition the space, and then associates each context with the partition defined by the closest centroid in cosine distance. Each partition thus defined gets a unique id, and word (X) – partition-id (Y) pairs are given to S-CODE as input. The algorithm cycles through the data until we get approximately 50 million updates. The resulting ϕ_x vectors are clustered using the k-means algorithm (no vector concatenation is necessary). Using default settings (64K random partitions, 25 s-code dimensions, $Z = 0.166$) the many-to-one accuracy is .7554 (.0055) and the V-measure is .6703 (.0037).

To analyze the sensitivity of this result to our specific parameter settings we ran a number of experiments where each parameter was varied over a range of values.

Figure 4 gives results where the number of initial random partitions is varied over a large range and shows the results to be fairly stable across two orders of magnitude.

Figure 5 shows that at least 10 embedding dimensions are necessary to get within 1% of the best result, but there is no significant gain from using more than 25 dimensions.

Figure 6 shows that the constant $\tilde{Z}$ approximation can be varied within two orders of magnitude without a significant performance drop in the many-to-one score. For uniformly distributed points on a 25 dimensional sphere, the expected $Z \approx 0.146$. In the experiments where we tested we found the real Z always to be in the 0.140-0.170 range. When the constant $\tilde{Z}$ estimate is too small the attraction in Eq. 6 dominates the repulsion in Eq. 7 and all points tend

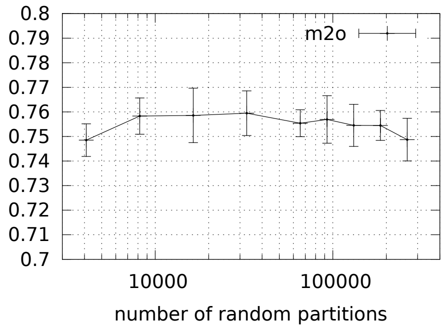

Figure 4
MTO is not sensitive to the number of partitions used to discretize the substitute vector space within our experimental range.

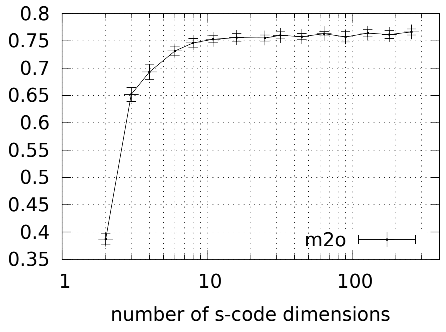

Figure 5
MTO falls sharply for less than 10 S-CODE dimensions, but more than 25 do not help.

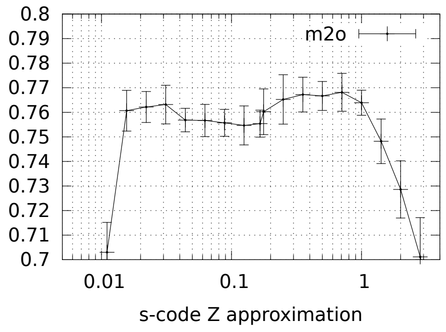

Figure 6
MTO is fairly stable as long as the $\tilde{Z}$ constant is within an order of magnitude of the real Z value.

to converge to the same location. When $\tilde{Z}$ is too high, it prevents meaningful clusters from coalescing.

We find the random partition algorithm to be fairly robust to different parameter settings and the resulting many-to-one score significantly better than the bigram baseline.

9.3 Random substitutes

Another way to use substitute vectors in a discrete setting is simply to sample individual substitute words from them. The random-substitutes algorithm cycles through the test data and pairs each word with a random substitute picked from the pre-computed substitute vectors (see Section 3). We ran the random-substitutes algorithm to generate 14 million word (X) – random-substitute (Y) pairs (12 substitutes for each token) as input to S-CODE. Clustering the resulting ϕ_x vectors yields a many-to-one score of .7680 (.0038) and a V-measure of .6822 (.0029).

This result is close to the previous result by the random-partition algorithm, .7554 (.0055), demonstrating that two very different discrete representations of context based on paradigmatic features give consistent results. Both results are significantly above the bigram baseline, .7314 (.0096). Figure 7 illustrates that the random-substitute result is fairly robust as long as the training algorithm can observe more than a few random substitutes per word.

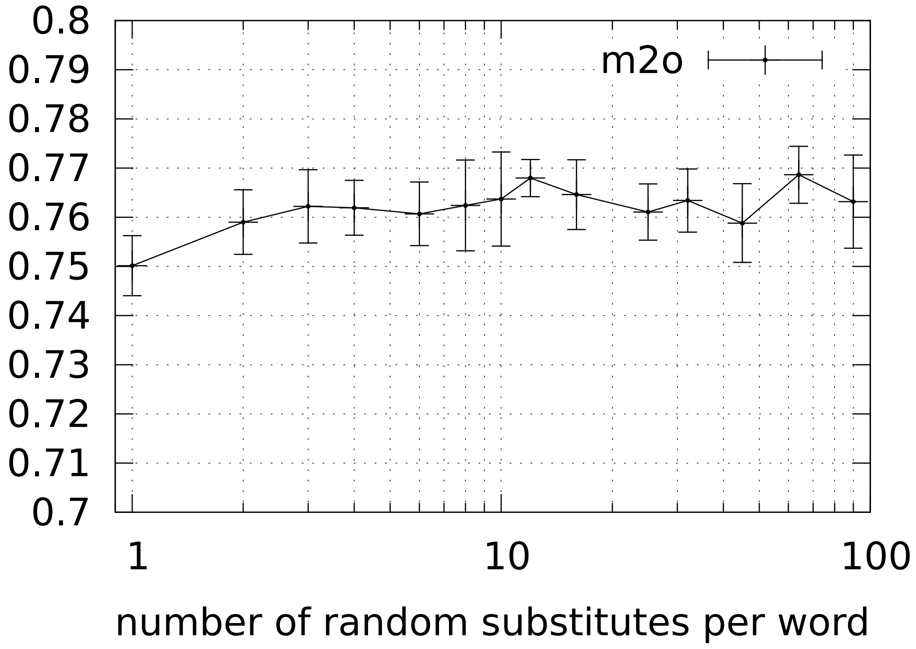

Figure 7
MTO is not sensitive to the number of random substitutes sampled per word token.

9.4 Morphological and orthographic features

Clark (**?**) demonstrates that using morphological and orthographic features significantly improves part-of-speech induction with an HMM based model. Section 2 describes a number other approaches that show similar improvements. This section describes one way to integrate additional features to the random-substitute model.

The orthographic features we used are similar to the ones in (**?**) with small modifications:

- Initial-Capital: this feature is generated for capitalized words with the exception of sentence initial words.

- Number: this feature is generated when the token starts with a digit.
- Contains-Hyphen: this feature is generated for lowercase words with an internal hyphen.
- Initial-Apostrophe: this feature is generated for tokens that start with an apostrophe.

We generated morphological features using the unsupervised algorithm Morfessor (**?**). Morfessor was trained on the WSJ section of the Penn Treebank using default settings, and a perplexity threshold of 300. The program induced 5 suffix types that are present in a total of 10,484 word types. These suffixes were input to S-CODE as morphological features whenever the associated word types were sampled.

In order to incorporate morphological and orthographic features into S-CODE we modified its input. For each word – random-substitute pair generated as in the previous section, we added word – feature pairs to the input for each morphological and orthographic feature of the word. Words on average have 0.25 features associated with them. This increased the number of pairs input to S-CODE from 14.1 million (12 substitutes per word) to 17.7 million (additional 0.25

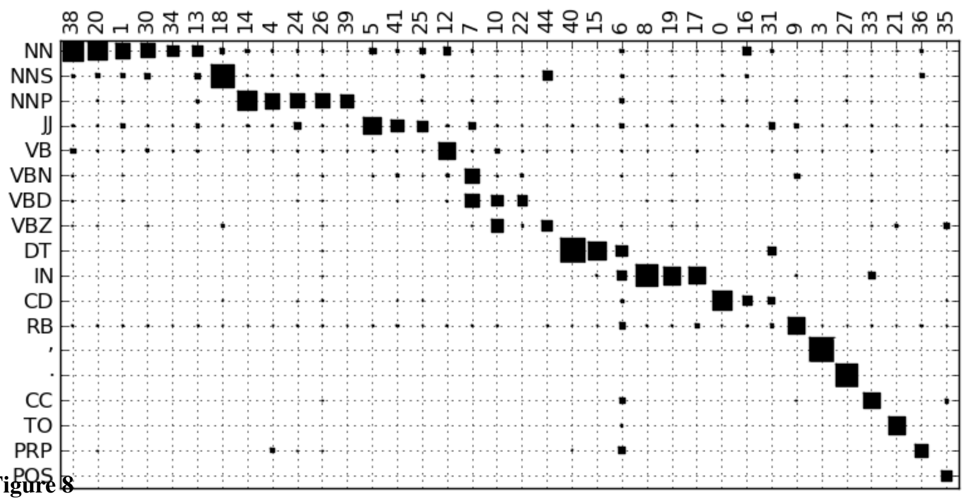

Figure 8
Hinton diagram comparing most frequent tags and clusters.

10. Multiple Languages

	Language	Types	Tags	k-means	SVD2	clark	BMMM	PYP	Best Published	Bigram	uPos	uPos+L	uPos+L+M
WSJ	English	49,190	45	.616 / .595	.640 / .582	.712 / .656	.728 / .661	.775 / .697	.761 / .688*	bigram	upos	-	.8004 / .7160
MULTEXT-East	Bulgarian	16,352	12+1	.593 / .503	.510 / .417	.665 / .556	.644 / .545	-	-	.6732 / .4119	.6883 / .5291	.7039 / .5496	.6754 / .5246
	Czech	19,115	12+1	.567 / .486	.509 / .355	.641 / .526	.642 / .539	-	-	.6269 / .4586	.6781 / .4829	.6742 / .4854	.6977 / .5042
	English	9,773	12+1	.654 / .565	.655 / .523	.706 / .605	.733 / .633	-	-	.7690 /. 6131	.8229 / .6610	.8282 / .6719	.8343 / .6787
	Estonian	17,845	12+1	.556 / .453	.553 / .387	.584 / .444	.644 / .533	-	-	.6089 / .4119	.6555 / .4437	.6634 / .4606	.6526 / .4418
	Hungarian	20,321	11+1	.539 / .467	.495 / .398	.614 / .489	.682 / .548	-	-	.6181 / .4514	.6914 / .5046	.7052 / .5244	.7287 / .5444
	Romanian	15,189	12+1	.551 / .452	.526 / .421	.499 / .409	.611 / .523	-	-	.6565 / .5202	.6469 / .5012	.6675 / .5269	.6488 / .5251
	Slovene	17,871	12+1	.562 / .469	.542 / .395	.694 / .549	.679 / .567	-	-	.6772 / .5044	.6873 / .5845	.6892 / .4901	.6833 / .4941
	Serbian	18,095	12+1	.470 / .414	.546 / .391	.641 / .510	.620 / .490	-	-	.6267 / .4510	.6240 / .4479	.6181 / .4421	.6336 / .4566
CoNLL06 Shared Task	Arabic	12,915	20	.607 / .433	.490 / .276	.598 / .406	.615 / .424	.675 / -	-		-	-	-
	Bulgarian	32,439	54	.656 / .536	.653 / .490	.704 / .596	.689 / .588	.732 / -	-	.6972 / .5532	.7399 / .5824	.7391 / .5856	.7207 / .5673
	Chinese	40,562	15	.611 / .326	.546 / .245	.567 / .318	.694 / .426	-	-			-	-
	Czech	130,208	12	-	-	.655 / .471	.657 / .484	.701 / -	-	.6944 / .5036	.6764 / .4867	.7149 / .5330	.6903 / .5227
	Danish	18,356	25	.616 / .517	.576 / .408	.653 / .527	.711 / .590	.762 / -	.667 / -†	.6757 / .5290	.7214 / .5559	.7520 / .5927	.7482 / .5958
	Dutch	28,393	13	.605 / .453	.524 / .367	.679 / .522	.711 / .547	.704 / -	.673 / -‡	.6703 / 5205	.7014 / .5405	.7393 / .5980	.7228 / .5925
	German	72,326	54	.675 / .587	.642 / .541	.739 / .630	.744 / .619	-	.684 / -‡	.7525 / .6285	.7637 / .6314	.7735 / .6554	.7529 / .6403
	Japanese	3,231	80	.762 / .761	.755 / .744	.774 / .786	.785 / .774	-	-	-	-	-	-
	Portuguese	28,931	22	.644 / .516	.631 / .459	.692 / .574	.768 / .639	.785 / -	.753 / -†	.7031 / .5617	.7381 / .5770	.7907 / .6317	.7948 / .6405
	Slovene	7,128	29	.642 / .526	.603 / .440	.635 / .539	.562 / .494	-	-	.6384 / .4976	.6503 / .4925	.6555 / .5036	.6572 / .5023
	Spanish	16,458	47	.692 / .595	.682 / .548	.719 / .616	.717 / .632	.788 / -	.732 / -†	.7086 / .5844	.7492 / .6983	.7718 / .6372	.7627 / .6331
	Swedish	20,057	41	.622 / .532	.591 / .474	.687 / .589	.682 / .580	.686 / -	.606 / -‡	.6721 / .5558	.6931 / .5654	.6946 / .5721	.6649 / .5613
	Turkish	17,563	30	.628 / .408	.524 / .274	.581 / .368	.587 / .402	-	-	.6069 / .3551	.6228 / .3804	.6348 / .4109	.65.00 / .4246
	French	49,964	23	.686 / .482	.685 / .463	.778 / .573	.766 / .550	-	-		-	-	-
	A.Greek	15,194	15	.448 / .386	.385 / .242	.454 / .333	.451 / .405	-	-		-	-	-

11. Error Analysis

Figure 8 is the Hinton diagram showing the relationship between the most frequent tags and clusters from the experiment in Section 9.4. In general the errors seem to be the lack of completeness (multiple large entries in a row), rather than lack of homogeneity (multiple large entries in a column). The algorithm tends to split large word classes into several clusters. Some examples are:

- Titles like Mr., Mrs., and Dr. are split from the rest of the proper nouns in cluster (39).
- Auxiliary verbs (10) and the verb "say" (22) have been split from the general verb clusters (12) and (7).
- Determiners "the" (40), "a" (15), and capitalized "The", "A" (6) have been split into their own clusters.
- Prepositions "of" (19), and "by", "at" (17) have been split from the general preposition cluster (8).

Nevertheless there are some homogeneity errors as well:

- The adjective cluster (5) also has some noun members probably due to the difficulty of separating noun-noun compounds from adjective modification.
- Cluster (6) contains capitalized words that span a number of categories.

Most closed-class items are cleanly separated into their own clusters as seen in the lower right hand corner of the diagram. The completeness errors are not surprising given that the words that have been split are not generally substitutable with the other members of their Penn Treebank category. Thus it can be argued that metrics that emphasize homogeneity such as MTO are more appropriate in this context than metrics that average homogeneity and completeness such as VM as long as the number of clusters is controlled.