On-Line Cumulative Learning of Hierarchical Sparse n-grams

Karl Pfleger [*]
Computer Science Department
Stanford University
kpfleger@cs.stanford.edu

Abstract

We present a system for on-line, cumulative learning of hierarchical collections of frequent patterns from unsegmented data streams. Such learning is critical for long-lived intelligent agents in complex worlds. Learned patterns enable prediction of unseen data and serve as building blocks for higher-level knowledge representation. We introduce a novel sparse n-gram model that, unlike pruned n-grams, learns on-line by stochastic search for frequent n-tuple patterns. Adding patterns as data arrives complicates probability calculations. We discuss an EM approach to this problem and introduce hierarchical sparse n-grams, a model that uses a better solution based on a new method for combining information across levels. A second new method for combining information from multiple granularities (n-gram widths) enables these models to more effectively search for frequent patterns (an on-line, stochastic analog of pruning in association rule mining). The result is an example of a rare combination—unsupervised, on-line, cumulative, structure learning. Unlike prediction suffix tree (PST) mixtures, the model learns with no size bound but using less space than the data. It does not repeatedly iterate over data (unlike MaxEnt feature construction). It discovers repeated structure on-line and (unlike PSTs) uses this to learn larger patterns. The type of repeated structure is limited (e.g., compared to hierarchical HMMs) but still useful, and these are important first steps towards learning repeated structure in more expressive representations, which has seen little progress especially in unsupervised, on-line contexts.

1 Introduction

For purposes here, *on-line learning* means the learner sees data a little at a time and cannot remember all data nor repeatedly iterate over it. This paper takes as a given the importance of on-line learning and will not dwell on motivating it. It suffices to note that (a) on-line learning is critical for long-lived autonomous agents in complex environments, (b) in many specific applications data sizes overwhelm storage capacities [1], and (c) a vast array of literature has argued for on-line learning (e.g., [1, 2, 3, 4]). To quote [2], "in a broad sense, online learning is essential if we want to obtain *learning* systems as opposed to merely *learned* ones." This paper further assumes the importance of *Cumulative learning* (also called layered or hierarchical learning), which involves using the results of prior learning to facilitate further learning (e.g., building new knowledge structures from experience by combining previously learned structures). For more discussion, see [5].

We present a model, *hierarchical sparse n-grams*, for on-line cumulative learning of frequently occurring patterns from unstructured unsegmented data, along with a related subcomponent model, *sparse n-grams*. Language, music, spatial configurations, event chronologies, action sequences, and many other types of data exhibit repeated substructure. For intelligent autonomous agents, identifying repeated substructure or frequent patterns in data has many benefits, including prediction of unseen information, improving short-term memory capacity and thus information processing capability generally, and facilitating communication and further learning. In addition, frequent patterns serve as building blocks for higher-level knowledge representation. General methods for identifying frequent patterns would greatly aid automated selection of higher-level representational units that are tuned to the environment. Frequent patterns in unbounded, unsegmented data streams can be identified on-line by simply noticing and remembering patterns that occur often, and using these to search for larger patterns that would otherwise have been more difficult to notice. Putting this into practice involves several challenges, as we will explain. We must emphasize that though our models are based on n-grams, our main goal is not model-class specific, but is the (under-investigated) on-line learning of frequent patterns in data and the cumulative use of existing patterns to help identify larger ones. Our purpose is not to tweak a slight performance improvement on

[*] The author performed this research at Stanford, but is presently affiliated with Google.

standard NLP, compression, or speech community n-gram benchmarks (most of which are batch and all of which are non-cumulative).

This paper is organized as follows. Section 2 presents sparse n-grams, the component model. This section demonstrates the benefits of the sparse representation, explains the method for on-line structure learning in the presence of sparseness, and introduces the mathematical form of novel probability estimates that form the basis for inference in both the basic and hierarchical models. Section 3 introduces hierarchical sparse n-grams and explains how the hierarchical nature of these models dramatically improves both the probability estimates (inference) and pattern selection (structure learning). Experimental results demonstrate that the models do an impressive job of finding frequent patterns demonstrated by their environments despite very sparse sampling of huge pattern spaces. Section 4 discusses related work from a diverse set of research communities. In general, the related models share certain properties but address different fundamental goals. The paper concludes with Section 5.

2 Sparse n-grams

n-grams are considered state-of-the-art for problems involving discrete sequences [6, 7]. Exhaustive n-grams store an occurrence count for every pattern of width n. We introduce *sparse n-grams* (or SNGs), which keep only some counts. They trade prediction quality for space, but wider SNGs can out-predict narrower exhaustive n-grams with the same number of patterns, making the sparse models useful when data is plentiful relative to storage. Similar n-grams trained exhaustively and then pruned [8] require increased storage during training and are not easily adaptable to on-line learning. Our goal is not merely to save space during training, however, or to improve n-gram models, but to investigate on-line, hierarchical learning of frequent patterns from unsegmented data. SNGs are a necessary subcomponent of the hierarchical models introduced below, but they are also interesting themselves and many issues are more easily introduced with them.

2.1 Sparse joint distributions and inference

For our purposes here, the learning and inference tasks are the unbounded, sequential analogs of those for standard fixed-feature IID unsupervised learning. A query is a sequence of any width with symbols for some positions specified, some targets to be predicted, and the rest missing. Each position is a symbol from alphabet A. n-grams keep a count C for each of the $|A|^n$ patterns and estimate probabilities as C/N (or a smoothed version, e.g., $\frac{C+1}{N+|A|^n}$), where N sums all counts. An SNG with counts for k patterns, called *tracked* patterns, for fixed k, is a k-n-gram. It estimates tracked patterns as above and distributes the remaining probability mass evenly among the untracked patterns to complete the joint distribution[1], which can be conditioned to make arbitrary predictions.

To demonstrate that sparseness can improve prediction, we trained sparse and exhaustive n-grams on book1 (a Thomas Hardy novel) of the Calgary corpus [7] (stripped of non-letters).[2] Batch training was used here just to demonstrate the potential of SNGs. We examined several inference patterns (forward, backward, and middle prediction and variants with missing values). Accuracy (0/1 loss) and cross-entropy were measured on a held-out test set of the last 10,000 characters. We expected sparseness to impair prediction, expected it to hurt cross-entropy more than accuracy (for which fine distinctions between unlikely events is less important), and expected increased width to mitigate the degradation.

Cross-entropy results were mixed, with n-grams outperforming SNGs in a few cases but not in others. Under accuracy, n-grams were always outperformed by a one-wider SNG with the same number of patterns (Figure 1(a)). This shows that the advantage of an extra predictor variable can outweigh the degradation caused by sparseness. Figure 1(b) shows that there is little degradation until the sparseness becomes severe. Also, any increase in storage can increase SNG performance by increasing k, whether or not it is enough to use a wider exhaustive model (increasingly important as n or $|A|$ rise). Since sparseness does not create inefficiencies[3], SNGs can be useful in some circumstances.

Sparseness degrades accuracy less than cross-entropy since accuracy depends only on correctness of the mode of the model's conditional distribution of the targets, not fine distinctions. The following is a more detailed explanation. When predicting with a full set of n–1 given symbols, the conditional distribution derives from renormalizing $|A|$ entries of the joint distribution. If a well-trained SNG includes at least one of these, it will get the correct mode. Only when *all* are absent will accuracy suffer, and this is the least likely case if the model includes frequent patterns.

[1]n-grams are often stored in a conditional form, but can be undirected. They are interchangeable for exhaustive but not sparse versions. We use the undirected here for simplicity, more flexible sparseness (see Section 4), predicting equally well backwards, and generalizability to 2D data (a long-term goal). ADtrees [9] allow efficient prediction even from the undirected representation.

[2]Our experiments have concentrated on letters as symbols (standard in the compression community [7]) instead of words. We believe the structure of letters, phonemes, musical notes, etc., is more appropriate for studying identification of frequent patterns (and might play a role in discovering words in the first place [5]). Words have longer-range interactions. Despite their word-level success, n-grams seem more suited to letters (also claimed by [10]). Nonetheless, our models are not limited to the letter level and should work well for word n-grams as well.

[3]Representation of SNGs uses a tree. Each leaf stores the count of a corresponding n-tuple ($O(n)$ lookup as for n-grams).

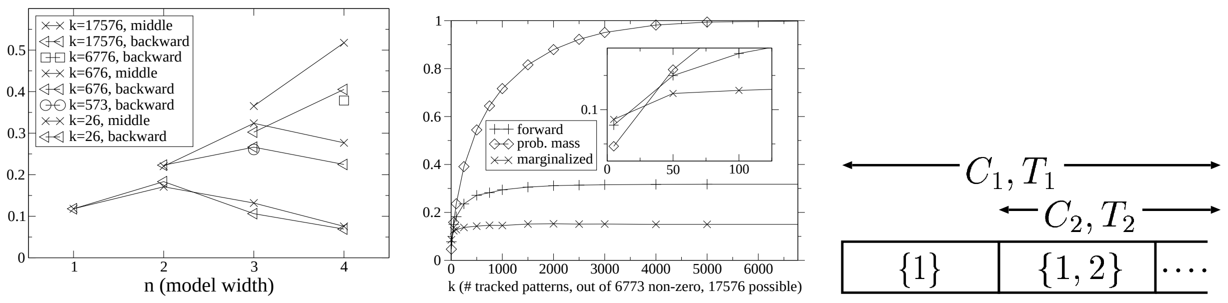

Figure 1. (a) k-n-gram accuracy vs. n for $k = 26^i$ (i=1–3) for backward and middle prediction. Forward behaves similar to backward and is omitted for clarity. Random guessing on this 26-class problem gets only 0.038 accuracy, and majority (a 1-gram) 0.118. The leftmost points of the equi-k lines are exhaustive n-grams, all outperformed by wider SNGs to the right. When k is limited to the number of non-zero counts in the exhaustive (n–1)-gram, performance (circle and square) still exceeds the exhaustive model. (b) k-n-gram accuracy (and total tracked probability mass) vs. k for n=3, for forward (`ggt`) and marginalized forward (`gmt`) prediction (Given, Missing, Target). Degradation is severe only for small k. (c) The counts kept.

Cross-entropy will degrade if *any* of these entries are replaced by the untracked average. When there are missing values, the conditional distribution derives from renormalizing sums of entries. Being larger, the tracked entries will usually dominate these sums so that if the model contains any of this much larger set of entries, accuracy will often not suffer. Indeed, Figure 1(b) shows that marginalized prediction degrades relatively less with sparseness. Under a more general loss function that penalizes incorrect answers but allows "skipping questions" for a smaller penalty, sparseness hurts even less since it is obvious to the model when it does not have the relevant counts to make a good guess. These concerns are important because accuracy (or more general loss functions allowing abstention) is more appropriate than entropy-based measures in some situations.

2.2 On-line learning of frequent patterns

The model cannot know the frequent patterns *a priori*, so on-line learning requires structural selection of which counts to include. Structure learning is harder on-line. Batch structure learning often works by optimizing a global metric such as a Bayesian posterior or MDL score over all data. This cannot be done on-line[4]. We identify frequent patterns on-line using a stochastic subset search that incrementally adds and removes patterns. Patterns are added randomly, but only when appearing in training (the add probability for a new instance is a small fixed value). Tracked patterns with low observed frequencies are discarded. Thus, infrequent patterns shuffle in and out of the model. The key that makes learning feasible is that frequent patterns, once added, are identifiable as such. A period of uncharacteristic rarity might cause a truly frequent pattern to be discarded, but if it is truly frequent then it will be added again, and as a pattern is tracked longer the chance of large enough anomalous drought to trigger removal becomes vanishingly small.

Addition and removal can operate independently or one can trigger the other to keep k fixed. If only the least frequent pattern is ever removed, the model will converge. Unfortunately, we do not know the true pattern frequencies but only slowly converging estimates. We created a new version of *Hoeffding races* [12, 1] to decide when to remove a pattern and which to remove. Maron and Moore [12] introduced Hoeffding races to the machine learning community for model selection in supervised learning, and Domingos and Hulten [1] used these races for on-line decision tree construction. We adapt the technique for sparse n-gram pattern removal and introduce a slight improvement specific to the low probabilities of this application.

Hoeffding races are useful when there are several competing entities about which statistical evidence is accumulating and the goal is to find an extreme (high or low) valued example. In the present context, this allows one to smoothly trade off how much less frequent the least-frequent pattern is with how converged the estimates are so that a bigger gap between the frequencies can be acted upon while the error bars are still high, while a close race will require more convergence. Traditional Hoeffding races use the Hoeffding bounds, a member of the broader class of Chernoff bounds. We use a tighter version of the Chernoff bounds [5, Sec.7.3.4], appropriate when the probabilities are close

[4]…unless the model encodes sufficient statistics to summarize past data, which is not possible with structure changes that grow new parameters. Sufficient statistics can be encoded where changes only shrink the models, as in [11] where segmented data allows direct data incorporation steps. Our techniques do not depend on segmented data. Note that adding growth steps (reversing bad merges) breaks the on-line nature of the algorithms in [11].

to zero, as they are for n-grams. The other difference from traditional Hoeffding races is that in this case the race is perpetual, with new competitors continually introduced after the race has started. The exact rule that sparse n-grams employ can be abstractly stated as follows: The model removes a pattern as soon as it can identify one that it is "reasonably sure" is "close" to being the least frequent (probably approximately the least frequent). Specifically, the model finds the pattern whose upper bound (highest possible probability in the confidence interval) is lowest. Then it finds a different pattern whose lower bound is lowest. If the difference between the upper bound of the former and the lower bound of the latter is less than a tolerance parameter τ, the model drops the former pattern. The continual addition and removal process thereby converges to a stochastic equilibrium in which more frequent patterns are more likely to be included.

For each pattern pat$_i$, the model maintains a count C_i of occurrences since inclusion. Since patterns are added at different times, empirical frequencies have different reliabilities and the model cannot simply use C_i/T as the estimate for every pattern, so it keeps the total number T_i of instances seen since adding pat$_i$. See Figure 1(c). C_i/T_i is less reliable for new patterns, so probability estimates must be weighted based on age to insure that after adding a new pattern, its probability under the model only slowly diverges from the untracked average $p_{\text{untracked}}$, asymptotically approaching C_i/T_i. To do this, the model keeps patterns sorted by T_i ($T_i > T_{i+1}$). For pat$_1$, $p_1 = C_1/T_1$, but p_2 has an additional term estimating missed occurrences from when only pat$_1$ was tracked, called episode$_1$: $p_2 = \frac{1}{T_1}\left[C_2 + (T_1 - T_2)(1-p_1)\frac{1}{|A|^n - 1}\right]$, where $(T_1 - T_2)(1-p_1)$ estimates the number of non-pat$_1$ occurrences in episode$_1$. Appealingly, this is a linear interpolation of $\frac{1-p_1}{|A|^n-1}$ (i.e., $p_{\text{untracked}}$ before adding pat$_2$) and $\frac{C_2}{T_2}$, weighted by T_2. The general formula adds terms for each episode:

$$p_i = \frac{1}{T_1}\left[C_i + \sum_{j=1}^{i-1}(T_j - T_{j+1})\cdot q_j \cdot \frac{1}{|A|^n - j}\right] \tag{1}$$

where $q_j \equiv (1 - \sum_{l=1}^{j} p_l)$. These approach C_i/T_i and can be computed in a linear sweep.

We have also investigated Bayesian and EM-based approaches to this problem. Bayesian predictions can be derived for a slight problem variant that assumes knowledge of the counts for each individual episode (a complex likelihood function involving nested interacting sums makes the original problem too difficult). It can be operationalized by approximating the count breakdowns by episode. An EM approach attempts to improve the uniform distribution on Equation 1's right side by instead using the resubstituted solution distribution. The resulting equations can be solved algebraically rather than requiring iteration to a fixed point. Under the modified problem, the solution is equivalent to the Bayesian solution. We implemented this and the above approach and in practice they behave almost identically. The next section improves on them. See [5, Sec.7.3.3] for more on these approaches and their relationships.

3 Hierarchical Sparse n-grams

Hierarchical sparse n-grams (HSNGs) consist of multiple sparse n-grams of consecutive widths (possibly an exhaustive n-gram as the smallest) and dramatically improve two aspects of the fixed-width models: the probability calculations and pattern selection. A single tree accommodates all patterns by storing counts in non-leaves for smaller-width patterns (still $O(n)$ lookup). This provides the same variance advantages as traditional multiwidth exhaustive n-grams but uses a new method for combining the models that is more elegant than linear interpolation or backoff models. Slowly growing versions of HSNGs can incrementally add greater widths during on-line training. These models smoothly "surf" the bias/variance curve by fitting parameters for ever-widening joint probability distributions, continually decreasing bias error (by widening the joint) as well as variance error (by converging on better parameter estimates).

3.1 Improved Probability Estimation

To estimate untracked occurrences, the EM approach above used the resubstituted width-n distribution, which *is unreliable precisely when needed, when too little data has been seen for the n distribution to have converged*. Smaller widths converge exponentially faster, so each level n in HSNGs uses the $n-1$ distribution to estimate the missing data. Equation 1 becomes:

$$p_{D_n}(\text{pat}_{n,i}) = \frac{1}{T_{\max}}\Bigg[C_{n,i} + \sum_{j=0}^{i-1}(T_{n,j} - T_{n,j+1})\cdot q_{n,j}\cdot p_D(\text{pat}_{n,i}|\neg\text{pat}_{n,1},\ldots,\neg\text{pat}_{n,j})\Bigg] \tag{2}$$

$p_{D_n}(\text{pat}_{w,i})$ is the probability of pat$_{w,i}$ (the i-th oldest width-w pattern) under the n distribution. $q_{l,j} \equiv (1 - \sum_{p=1}^{j} p_{D_l}(\text{pat}_{n,p}))$, is as before the rest of the probability mass without the j oldest patterns (at level l). For HSNGs, $D = D_{n-1}$ instead of D_n (the EM approach) or uniform D_0 (Equation 1). The conditioning is just renormalization:

$$p_{D_{n-1}}(\text{pat}_{n,i} \mid \neg\text{pat}_{n,1},\ldots,\neg\text{pat}_{n,j}) = \frac{p_{D_{n-1}}(\text{pat}_{n,i})}{q_{n-1,j}} \tag{3}$$

This all reduces to Equation 1 in the base case.

Of course, the n–1 distribution does not specify n-tuple probabilities directly. The trick is to use estimates based on the order-(n–1) Markov assumption:

$$\begin{aligned} &P(x_1, x_2, \ldots, x_n) \\ &\quad= P(x_1, x_2, \ldots, x_{n-1})\cdot P(x_n \mid x_1, x_2, \ldots, x_{n-1}) \\ &\quad\approx P(x_1, x_2, \ldots, x_{n-1})\cdot P(x_n \mid x_2, \ldots, x_{n-1}) \\ &\quad\approx \frac{P(x_1, x_2, \ldots, x_{n-1})\cdot P(x_2, \ldots, x_{n-1}, x_n)}{P(x_2, x_3, \ldots, x_{n-1})} \end{aligned} \tag{4}$$

The approximation from the first to second lines above, $P(x_n \mid x_1, x_2, \ldots, x_{n-1}) \approx P(x_n \mid x_2, \ldots, x_{n-1})$, is in a sense the best estimate given only order-(n–1) knowledge. This gives us a way to effectively "widen" an n-gram, using two lookups at width $n-1$ and one lookup at width $n-2$ to determine a width-n probability.

For untracked patterns ($i > k_n$, where k_n is the number of width-n patterns), width n falls back on width n–1 directly (and renormalizes):

$$p_{D_n}(\text{pat}_{n,i}) = p_{D_{n-1}}(\text{pat}_{n,i}) \frac{q_{n,k_n}}{q_{n-1,k_n}} \quad (5)$$

Using Equation 1, the probabilities for all patterns can be computed in a linear sweep. This works despite that each pattern's probability involves the sum of a linear number of terms since the terms are the same from one pattern to the next, but that is not true of Equation 2 due to the rightmost term. Thus, straightforward computation of all probabilities for a level would require computation quadratic in the number of patterns. However, using the expansion of Equation 3 the portion of the rightmost term that changes from pattern to pattern can be factored out, preserving the ability to compute probabilities for all patterns in the model in a linear sweep.

With cached probabilities for the tracked patterns, looking up the probability for an untracked pattern requires three lookups to narrower submodels using Equations 4 and 5. Any of these may in turn be absent, requiring three more lookups. Nonetheless, querying the widest model for a specific probability takes $O(n^2)$ rather than time exponential in n since some of the submodel lookups can be handled by the same traversals down the tree. (Lookup requires up to n tree traversals, each of at most n steps.) n is typically very small relative to the overall size of the model. Furthermore, many inference patterns that require lookup of several patterns can be handled more efficiently since many patterns will be accessible from one traversal. (E.g., looking up all possible single-symbol extensions of a pattern requires only one traversal from the root.) This is a worst case. Normally, each narrower level contains more of the necessary patterns, making typical lookups linear in n. Other optimizations are also possible (see [5, Sec.8.2.2] for further optimizations or more detailed derivations of the equations here.)

Predictions are made using the largest width. Lower-order information filters up through the submodel calls. Smaller widths have larger Ts and more fully converged estimates. Dynamically adding a new level will not jar the distribution as a model having mostly small T values will smoothly fall back on the robust narrow information except where it identifies strong wide patterns in the data. Estimates both within and across widths are naturally weighted by their reliabilities. Authority smoothly transitions to greater widths as more data is seen.

3.2 Improved Pattern Selection

Fixed-width models must search a large space blindly. HSNGs use smaller known-frequent patterns to bias new

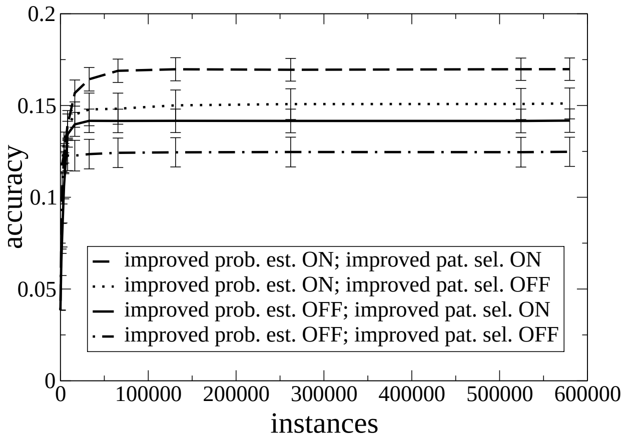

Figure 2. Learning curves for a sparse 2-gram using a pre-trained 1-gram submodel for probability estimation, pattern selection, both, or neither on book1. Each submodel use yields benefit.

selections. Specifically, the probability for adding any new n-tuple, rather than a small constant, is proportional to its estimate based on D_{n-1}. This can be viewed as a refinement of a simple strategy of combining existing frequent patterns to yield larger likely-frequent patterns. This refinement automatically takes into account all of the possibly many ways to parse the new pattern into existing patterns and the frequency estimates for all of these existing patterns. Since each level seeks frequent patterns, a bias based on their probabilities functions as a bias for compositional chunking of known patterns. This is the key to the cumulative aspect of HSNG learning—the learning at narrower levels is not only combined with the results of learning at the next wider level for better prediction (as in traditional multiwidth n-gram combinations) but also *directly enables* the structure learning at this wider level, which is critical as we will see in the results below.

HSNGs can use fixed k_ns (the number of patterns at each level) or the k_ns and number of levels can grow slowly as more data is seen by applying the add and remove rules independently and always considering addition of a new widest pattern (which creates a new level). Since wider patterns generally have lower absolute probabilities and vastly more patterns, higher levels converge more slowly.

3.3 Results

Figure 2 shows that each use of submodels above provides significant benefit (which can be greater when combined). Similar results are seen for larger n but with the improvement from probability estimation being more dominant when used with fully-converged submodels (because the widening approximation becomes more accurate for larger n). To test frequent pattern identification, hierarchical models were trained on several large datasets using slowly growing k_ns (and numbers of levels). The following tables shows k_n and the top 5 patterns by model probability for each n after training on 30 million letters of Reuters newswire text (1994 section of North American News corpus with non-letters removed.)

n	2	3	4	5	6
k_n	367	1540	1131	433	142
	th	the	tion	ation	nation
	he	ing	said	ofthe	saidth
	in	and	nthe	inthe	ingthe
	er	ion	ther	tions	ations
	an	ent	dthe	ingth	aidthe

n	7	8	9
k_n	38	11	7
	saidthe	official	president
	esident	resident	ternation
	residen	presiden	residento
	nationa	thenatio	theminist
	preside	tsaidthe	andforthe

The learned chunks are reasonable substrings for this text. With spaces removed, frequent word, sub-word, and super-word patterns were all mixed together. The following table shows how many of the most frequent 100 and 1000 patterns (by true corpus frequency) were learned by the model for the smaller widths (larger widths were still far from converging).

	Reuters			
n	2	3	4	5
k_n	367	1540	1131	433
top100	100	100	99	75
top1000	n/a	910	560	238
unique	670	12556	121799	626465

Similar results were achieved with speech data (the TIMIT corpus) and DNA data (chromosome 22 of the human genome, about 34 million base pairs). The DNA results are shown below. Note that our purpose in testing on different data types was not to show particular performance on problems of traditional importance in each of these domains but to demonstrate that our technique for learning compositional patterns and estimating their prevalence is general enough to handle several different kinds of data.

	chromosome 22						
n	5	6	7	8	9	10	11
k_n	886	2094	1629	737	279	114	55
top100	100	100	99	68	44	28	19
top1000	886	991	669	258	128	57	32
unique	1024	4096	16384	65536	261726	1.0mil	3.5mil

The models do an excellent job of identifying the frequent patterns. For Reuters $n = 5$, for example, the model had added 434 patterns and removed 1 (it was still early in convergence at this level), but included 75 of the top 100 out of 626,465 unique 5-tuples that occurred (out of 11.8 million possible)—a dramatic improvement over chance guessing. Further, over half of the patterns added at the 5 level are in the top 1000 (top 0.16%). This is even more impressive considering the $n = 4$ level contained fewer than 1% of the data's unique 4-tuples. This demonstrates that the narrower submodels do an impressive job enabling successful pattern selection at the wider levels (just as those levels will in turn do for even wider levels).

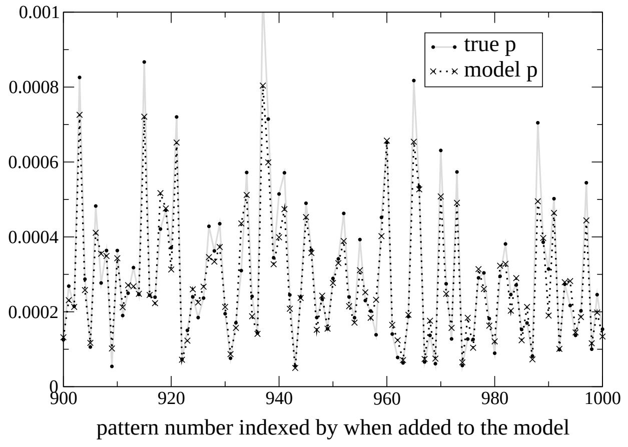

Figure 3. Comparison of the Reuters-trained hierarchical model's estimate vs. the true probability for the 900-1000th oldest 3-grams. The model tracks true probabilities well.

We examined how well the model's probabilities matched true corpus frequencies. Standard summarizations such as relative entropy of the distributions are inappropriate for the sparse representation. Also this metric is dominated by errors in low probabilities, which are specifically sacrificed by these sparse models. Thus, we just inspected all the values directly, and all but the very newest patterns matched quite well. Figure 3 shows a small sample section of patterns. The majority of patterns were older than the sample shown and matched even better than those shown.

4 Related Work

Pruning already-batch-trained n-grams has been investigated [8]. Our sparseness is similar to count cutoffs (widely used in practice). Our models, however, grow rather than scale back their storage and never need excessive space during training. That literature also discusses tradeoffs between space and predictive performance, but this research is the first we are aware of to explicitly address the tradeoff between memory reduction and increased predictive accuracy from increased n and the first to explain that accuracy (0/1 loss) suffers less from missing counts than cross-entropy.

Prediction suffix trees (PSTs) [10, 13, 14] are multi-width sparse models based on unbalanced trees. Each node represents a string and stores a conditional distribution for the next symbol given the string as preceding symbols. PSTs are grown by adding nodes that (a) represent frequent strings and (b) have distributions sufficiently different from their parent's or have a descendant with this property. Our models differ in several ways. PST learning contains forward-directed inductive bias since it chooses patterns in terms of forward distributions. Thus, PSTs do not make equally good use of information on both sides of a target. Second, our undirected representation allows for greater, more flexible sparseness. PSTs store a full conditional distribution at each node. HSNGs store one probability. A PST with a depth-d node must store counts for all $|A| \quad (d{+}1)$-tuple extensions, while an HSNG can represent any subset of these. The counts saved can be used elsewhere, improving other predictions. Most importantly, PST learning does not utilize repeated substructure. It is not more likely to extend a model by existing sequences with high counts from elsewhere in the tree. PST mixtures [14] use incremental updates, but not on-line structure learning in our sense. Either they are given a depth bound and add all sufficiently small strings that occur, or they have no bound and add all strings that occur. The latter simply reorganizes and remembers all data (often impractical). In the former case, the model cannot grow to recognize patterns wider than the bound and may require memory exponential in the bound. We desire a middle ground where the model grows *slowly*, but *without bound*. A model should be able to include any pattern demonstrated strongly enough by the data.

ADtrees [9], structures for efficient access to discrete multivariate data, are both similar to SNGs and complementary in that they can speed up SNG inference. ADtrees trade space for speed while SNGs trade prediction performance for space. The combination can balance all three. In addition, ADtrees use a sparse representation synergistic with that of SNGs. Applying ADtrees to SNGs allows much greater ADtree pruning (more speed for less space) when compared with application to exhaustive n-grams since all untracked patterns can be pruned.

Models equivalent to SNGs were used as hierarchical Bayesian priors [15]. We have moved the sparseness from the prior to the model itself.

Using the submodel to bias pattern selection is analogous to pruning in association rule mining [16, Section 20.6.2] [17] and sequence mining (e.g., [18]). Our heuristic is on-line and stochastic rather than absolute. Since it does not apply the same threshold to each level, it is more flexible. It seeks the most frequent patterns at each level regardless of their absolute frequencies. Neither the basic pruning ideas nor the related incremental association rule work (based on a very different problem with different assumptions from our work) make clear how to do the job done by HSNGs. For HSNGs, no pattern at the next level is prunable. They must rely on a probabilistic interlevel bias.[5]

Our pattern selection bias (that amounts to a bias for combinations of existing frequent patterns) is similar to the compositional chunking of Sequitur [19] and similar systems, but with finer statistical sensitivity than Sequitur, which is very greedy. Also note that Sequitur remembers all of its input in order to chunk. Therefore, it is not a true on-line algorithm in the sense used here.

Maximum entropy (ME) [6, 20] (and related NLP) techniques use "features" more general than our patterns. Uniform completion of the distribution in SNGs is consistent with ME. In ME, features are added incrementally in a process seemingly similar to ours but actually very different. Each addition requires iterating over all training data. ME, transformation-based techniques [6], and related schemes all require batch training and it is not clear how to adapt them for on-line learning, though our models may suggest some necessary ingredients. ME uses its features as constraints, but it fundamentally assumes all constraints are equally statistically reliable (fine in a batch context but not when new features are introduced after different amounts of data). ME would overweight recently introduced frequency estimates. Some sort of regularization, as introduced here, is required.

Hierarchical HMMs can represent sparse collections of patterns in a more expressive representation, but only recently has structure learning been addressed [21] and only for batch training. Even with complete random access to the data, identification of repeated substructure is problematic. There are other learning models that representationally subsume our models, but in each case the learning problem they address is dramatically different due to use of a domain theory, supervision, batch training, or structured or segmented data (usually many of these, see [5, Sec.8.3, Sec.10.4, Ch.9]). In no case can they accomplish the learning described here.

5 Conclusion

Hierarchical sparse n-grams can be viewed as combining multiwidth n-grams from the NLP and text compression communities, frequent itemset pruning from the KDD community, and on-line learning in the ML and connectionist traditions. (This suggests much cross-fertilization. E.g., interlevel flow of information to help direct search is a key thing lacking from PST work.) The difficulties in combining sparseness with on-line learning are fundamental and require more than subtle changes to exhaustive batch techniques.

Sparse n-grams are useful alternatives to exhaustive n-grams when data overwhelms memory. For autonomous agents in complex environments, this is always the case in

[5]The width-n distribution provides no useful absolute bounds on $n+1$ probabilities. If there were a low-frequency n-tuple, the model could avoid its extensions, but the sparse models by their nature do not retain confidently estimated low-frequency patterns.

the long run. Hierarchical sparse n-grams learn frequent patterns using fewer parameters than the number of potential patterns and without remembering all data or repeatedly iterating over it. They use use novel techniques for falling back on narrower distributions. A stochastic bias for compositions of smaller known-frequent patterns facilitates on-line learning of increasingly complex sparse representations. The result is a model capable of finding frequent patterns in huge search spaces and cumulatively constructing ever-larger representations of the frequent patterns demonstrated by the environment.

We have necessarily concentrated on prediction as the primary way to utilize knowledge of frequent patterns (or chunks), but frequent chunks can be very valuable in many other ways within a larger computational system. Learned chunks can act as aids to increase working memory capacity, based on substitution recoding, which improves information processing capacity quite broadly.[22] Frequent patterns are important for developing communication or shared language. Frequent chunks can serve as important features for other types of learning and can enable the automatic formation of associations that would otherwise be impossible to induce. For more on each of these topics, see [5, Sec.10.3]. The point is that a single general learning mechanism can build representations that both enable useful predictions and also serve as foundations for improving several aspects of an agent's cognitive behavior.

Sparseness is fundamental in complex knowledge representation. Whether in semantic networks, frames, description logics, ontologies, etc., knowledge is always non-exhaustive for real-world domains. We must continue to investigate domain-independent techniques for choosing which patterns exhibited by the environment to include in sparse representations. For more extensive treatment of all material, including motivations, derivations, a formal description of the learning problem, experimental results, related work comparisons, and explanations of the usefulness of frequent patterns in the future research landscape, see [5].

References

[1] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Knowledge Discovery and Data Mining*, pages 71–80, 2000.

[2] Richard Sutton and Steven Whitehead. Online learning with random representations. In *Proceedings of the 10th International Conference on Machine Learning*, 1993.

[3] Paul E. Utgoff. Many-layered learning. *Neural Computation*, 2002. In press.

[4] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504), 2000.

[5] Karl Pfleger. *On-Line Learning of Predictive Compositonal Hierarchies*. PhD thesis, Stanford University, June 2002. See http://ksl.stanford.edu/~kpfleger/thesis/.

[6] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. 1999.

[7] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice-Hall, 1990.

[8] Joshua Goodman and Jianfeng Gao. Language model size reduction by pruning and clustering. In *International Conference on Spoken Language Processing*, 2000.

[9] Andrew Moore and Mary Soon Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1998.

[10] H. Schutze and Y. Singer. Part-of-speech tagging using a variable memory Markov model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1994.

[11] A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, Berkeley, 1994.

[12] Oded Maron and Andrew Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing*, 1993.

[13] D. Ron, Y. Singer, and N. Tishby. The power of amnesia. *Machine Learning*, 25, 1996.

[14] Fernando Pereira, Yoram Singer, and Naftali Z. Tishby. Beyond word n-grams. In *Proceedings of the Third Workshop on Very Large Corpora*, 1995.

[15] Nir Friedman and Yoram Singer. Efficient Bayesian parameter estimation in large discrete domains. In *Advances in Neural Information Processing Systems*, 1998.

[16] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Prentice Hall, 2002.

[17] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[18] H. Mannila, H. Toivonen, and A. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1995.

[19] Craig G. Nevill-Manning and Ian. H. Witten. Identifying hierarchical structure in sequences: a linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.

[20] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), 1997.

[21] Kevin Murphy and Mark Paskin. Linear time inference in hierarchical HMMs. In *Advances in Neural Information Processing Systems 14*, 2002.

[22] George A. Miller. The magic number seven, plus or minus two: Some limits of our capacity for processing information. *Psychological Review*, 63, 1956.