Probabilistic Grammars and Languages

András Kornai
MetaCarta Inc.
350 Massachusetts Ave
Cambridge, MA 02139
andras@kornai.com

Abstract

A classic theorem of Ellis (1970) asserts that there are probabilistic languages (p-languages) that are not generated by any probabilistic finite state grammar (PFSG) even though their support (the set of strings with nonzero probability) is regular. Ellis' proof is highly algebraic, and gives no indication *why* such p-languages exist. Suppes (1970) argued that Ellis' construction is an artefact of using irrational probability values: since the probabilities that matter all go back to frequency counts i.e. rational numbers, he conjectured that for every p-language L with type i support there must be a probabilistic grammar (p-grammar) of the same type that generates a language L' that is statistically indistinguishable from L. Here we prove a strengthened version of the Suppes conjecture: for any p-language L with any type $i \leq 3$ support and for arbitrary $\epsilon > 0$ there is a probabilistic finite state language (PFSL) L' that approximates L within ϵ. While the result seemingly justifies Suppes' point of view, we argue that in the final analysis it is Ellis who had this right. We provide an asymptotic characterization of PFSLs over a one-letter alphabet in terms of exponentially decaying probabilities: this characterization makes it clear that only a small subclass of p-languages with regular support behaves the way we expect from languages generated by PFSGs and further, that the matter is quite independent of whether the probability values are rational. Having a more nuanced understanding of the mathematical relation between p-grammars and p-languages puts us in a better position to address the practical problems of fitting PFSA to real-world data, and here we take the first steps in this direction by discussing why Hidden Markov Models, which offer only a slight reformulation of PFSA, escape the exponential decay problem and offer a way out of the poverty of stimulus issue first noted by Miller and Chomsky (1963).

1 Background

A type i probabilistic grammar (or p-grammar for short) is standardly defined (see e.g. Levelt 1974) as a type i grammar where all productions have some nonzero probability – if the sum of probabilities assigned to productions sharing the same left hand side is 1 we speak of *normalized* p-grammars. The probability of a derivation is the product of the probabilities of the rules that appeared in the derivation (with multiplicity), and the probability of a string is the sum of the probabilities of its derivations. As is well known, the fact that a p-grammar is normalized is insufficient to guarantee that the probabilities it assigns will sum to 1. For example, the CFG that generates all binary branching structures, $S \rightarrow SS|a$, when both rules are applied with $p = 0.5$, will yield weights that sum to 0.5.

A probabilistic language (or p-language) is a language L and some assignment $p : L \rightarrow \mathbb{R}^+$ such that the sum of the assigned values is 1. In the finite state domain the normalization restriction is sufficient: normalized probabilistic finite state automata (PFSA) generate strings with probabilities that sum to 1. Since we could always extend p to assign 0 to all strings not in L, it is generally sufficient to speak of functions $f : T^* \rightarrow \mathbb{R}_0^+$ for which $\sum_{\alpha \in T^*} f(\alpha)$ is convergent and identify the p-language L as the *support* of f i.e. the set of strings α for which $f(\alpha) > 0$. Given that a p-grammar is simply a grammar plus an assignment of probabilities, and a p-language is simply a language plus an assignment of probabilities, it is reasonable to have the following

Conjecture 1 Type i p-grammars correspond to type i p-languages.

In one direction, the proof is relatively easy: ignoring the production probabilities a type i p-grammar G will, by definition, yield a type i language L, and it is clear that all and only strings in L will have positive derivation probability. If we assign the start symbol $S \in T$ the value 1, and make an appeal to the fact that the p-grammar is normalized, it is easy to see that the total probability mass assigned to sentential forms reachable from S in n steps is always 1, partitioned so that terminal strings so reachable have t and those with nonterminals still present have $1 - t$. By a limit argument it follows that the total probability mass assigned to members of L will be ≤ 1, and if it is strictly less than 1 we can always add in some extra string to absorb the difference.

In the other direction, however, the going is not so smooth: even over a one-letter alphabet, and already for type 3 languages, we can find a p-language L_3 for which no p-grammar (or p-acceptor) will ever work. The first such example was constructed by Ellis (1970), who focused on the type 3 case: here we present a version of his proof that extends his result to type 2 grammars (and can be pushed further, but this would take us away from the main line of argumentation). Note that over a one-letter alphabet $T = \{a\}$, by assigning each string a^n a weight $f(a_n) = p_n > 0$ such that $\sum_{i=1}^{\infty} p_i \leq 1$, and possibly padding out this sum by assigning $p_0 = 1 - \sum_{i=1}^{\infty} p_i$ to the empty string, we can guarantee that the support language, the set of all strings with or without the empty string, is regular.

Theorem 1 PCFGs over a one-letter alphabet do not generate all one-letter p-languages with regular support.

Proof Define the weights as a set with infinite transcendence degree over $\mathbb{Q}$. (That such sets exist follows from the fundamental theorem of algebra and from the existence of irreducible polynomials of arbitrary degree. If $t_1, t_2, t_3, \ldots$ is such a set, so will be $s_1, s_2, s_3, \ldots$ where $s_i = |t_i|/(1 + |t_i|)2^i$, and the latter will also sum to ≤ 1). Now consider the generating functions which are defined by taking the nonterminals as unknowns and the terminal a as a variable in the manner of Chomsky and Schützenberger (1963), except using the probabilities assigned to the rules as weights: for example the grammar of binary trees used above yields the functional equation $S = 0.5S^2 + 0.5a$. In the general case, solving the set of equations for the generating function associated to the start symbol S is very hard, but over a one-letter alphabet the only variable introduced by CF rules will of necessity commute with all probabilistically weighted polynomials in the same variable. Since all defining equations are polynomial, the output probabilities are algebraically dependent on the rule probability parameters. Since a CF rule system with n rules can generate at most n algebraically independent values, it follows that $s_1, s_2, s_3, \ldots, s_{n+1}$ can't all be obtained from the CFG in question. □

Discussion The original proof by Ellis constructs an infinite set of weights as $1/\sqrt{p_i}$, where p_i is the smallest prime larger than 4^i, and proceeds by counting the degree of the field extensions. This works well for the finite state case, where the functional equations are linear and the solutions are rationally dependent on the rule probabilities, but for CF and more complex grammars the equations are polynomial and only algebraic dependence can be assumed.

Thus, *there are more things in probabilistic heaven and earth than are dreamt of in your grammatical philosophy*. In the regular domain already, there are p-languages outside the reach of regular or even context free p-grammars. The situation is made particularly nasty by demonstrating the phenomenon over a one letter alphabet: clearly if we find grammatically unreachable probability distributions over this alphabet we can create examples over any size alphabet whatsoever. In response, Suppes (1970) argued that

> From the empirically oriented standpoint (...) Ellis' example, while perfectly correct mathematically, is conceptually unsatisfactory, because any finite sample of L drawn according to the density p could be described also by a density taking only rational values. Put another way, algebraic examples of Ellis' sort do not settle the representation problem when it is given a clearly statistical formulation. Here is one such formulation. (...)
>
> *Let L be a language of type i with probability density p. Does there always exist a probabilistic grammar G (of type i) that generates a density p' on L such that for every sample s of L of size less than N and with density p_s the null hypothesis that s is drawn from (L, p'_s) would not be rejected?*
>
> I have deliberately imposed a limit N on the size of the sample in order directly to block asymptotic arguments that yield negative results.

Suppes conjectured that the problem, stated thus, has an affirmative solution, and Levelt (1974:44) actually lists this as a theorem (though the surrounding text makes its conjectural status quite clear):

Conjecture 2 p-languages with type i support are statistically indistinguishable from languages generated by type i p-grammars.

In Section 2 we will investigate whether this Suppes/Levelt conjecture is tenable by explicitly characterizing the languages that can be generated by type 3 p-grammars or, what is the same, by PFSA, over a one letter alphabet. Our main result is Theorem 2, which asserts that PFSA, (possibly also endowed with silent λ-moves that change state but emit no symbol) generate all and only p-languages with probabilities that show exponential decay. Next we show that Conjecture 2 can be strengthened (p-languages with type i support are statistically indistinguishable from languages generated by PFSA) but only in a trivial sense: for more ambitious language modeling efforts Theorem 2 still presents an insurmountable barrier. This is somewhat surprising in light of the fact that finite state language modeling, in the form of Hidden Markov Models (HMMs) remains, to this day, the dominant algorithm in computational linguistics – we investigate the reasons for the disconnect between theory and practice in the concluding Section 3.

2 PFSA over a one letter alphabet

To define a *probabilistic finite state automaton (PFSA)* over a one-letter alphabet $\{a\}$ requires a set of states $\Sigma = \{s_1, \ldots, s_n\}$, for each state i a set of values $t_{i,j}$ that characterizes the probabilities of moving from state s_i to s_j upon consuming (emitting) the letter a, and a set of values $l_{i,j}$ that characterizes the probabilities of moving from state s_i to s_j by λ-move i.e. without consuming (emitting) a. In what follows we will treat PFSA as generating devices – the results presented here remain true for acceptors as well. To simplify the notation, we add a start state s_0 which only has λ-transitions to s_i for $i > 0$, and replace all blocked transitions by transitions leading to a sink state s_{n+1} that has all (emitting and non-emitting) transitions looping back to it and has weight 0 in the vector $\mathbf{w}$ that encodes the mixture of accepting states. This way, we can assume that in every state s_i and at every time tick the automaton A will, with probability 1, move on some state s_j and emit (or not emit) a symbol during transition with probability $t_{i,j}$ (resp. $l_{i,j}$) collected in the matrix T (resp. L). We have the following

Theorem 2 If $p : \{a\}^* \to \mathbb{R}^+$ is a PFSA p-language it is ultimately periodic in the sense that there exists a fixed k and l such that for all $0 \leq i < k$ either all weights $p(a^{i+rk})$ are zero once $i + rk > l$ or none of the weights $p(a^{i+rk})$ are zero for any r such that $i + rk > l$ and all weight ratios $p(a^{i+rk+k})/p(a^{i+rk})$ tend to a fixed value $\lambda_1^k < 1$.

Proof We only sketch the proof here – for details see Kornai (2007) Theorem 5.8. The probability $P(a^k|A)$ of A emitting a^k is the sum of the probabilities over all paths that emit a k times. We add a zero symbol z and consider the automaton A' that emits z wherever A made a λ-transition: we collect the probabilities in a formal power series $p(a, z)$ with non-commuting variables a and z – in matrix notation, $p(a, z) = \sum_{k \geq 0}(aT + zL)^k$. Given a fixed start state s_0 and some weighted combination $\mathbf{w}$ of accepting states, the probability of a string $x_1, \ldots, x_n \in \{a, z\}^n$ being generated by A' is obtained as the inner product of the zeroth row of $(T + L)^n$ with the acceptance vector $\mathbf{w}$. The spectral radius of L is less than 1 (since states with no emission ever can be eliminated) and the matrix series $I + L + L^2 + L^3 + \ldots$ converges to $(I - L)^{-1}$. This gives

$$P(a^k|A) = \mathbf{e}L((I - L)^{-1}T)^k(I - L)^{-1}\mathbf{w} \tag{1}$$

Since the only parts of (1) dependent on k are the $k-$th powers of a fixed matrix $(I - L)^{-1}T$, the growth of $P(a^k|A)$ is expressible as a rational combination of k-th powers of constants $\lambda_1, \lambda_2, ..., \lambda_n$ (the eigenvalues of $(I - L)^{-1}T$ arranged in decreasing order) with the fixed probabilities $t_{i,j}$ and $l_{i,j}$. Therefore, the probabilities $P(a^n|A)$ and $P(a^m|A)$ will be both dominated by a rational function of λ_1^n (resp λ_1^m) (recall that by the Perron-Frobenius theorem λ_1 will be real, positive, unique, strictly greater in absolute value than all other λ_i, and strictly less than 1 for each connected component) so their ratio will tend to λ_1^{n-m}. (Because of multiple transitive components connected by one-way λ-transitions linear multiplicative terms can accompany the exponential main term, but these tend to 1 as we take the limit of the ratio.) □

Example 1 Let $p_0 = 2^{-1}, p_1 = p_2 = p_3 = p_4 = 2^{-2}/2^{2^1}, p_5 = \ldots = p_{20} = 2^{-3}/2^{2^2}$ and in general divide the probability mass 2^{-n} among the next 2^{2^n} strings. By Theorem 2 this distribution will differ from any distribution that is obtained from PFSA by inspecting frequencies is finite samples, even though all components are rational, since in PFSA log probabilities can change at most linearly, rather than exponentially as the example demands.

Theorem 2 provides, and Example 1 exploits, exactly the kind of asymptotic characterization that Suppes wanted to avoid by limiting attention to samples of a fixed size $< N$. In hindsight, it is easy to see where the strict empiricism embodied in Conjecture 2 misses the mark: with the availability of corpora (samples) with $N > 10^{10}$ (bigger than the entire corpus available to language learners in the first twenty years of their lives) it is evident that our primary goal is not to characterize the underlying distribution to 10 significant digits, but rather to characterize the tail, where probabilities of 10^{-40} or many orders of magnitude below are quite common. Since perfectly ordinary words often have frequencies below 10^{-6} or even 10^{-9}, rather short sentences containing these, e.g. *In our battalions, dandruffy uniforms will never be tolerated* will have probability well below 10^{-40}. There is no surprise that perfectly grammatical sentences can have extremely low probability (see Pereira 2000) – the primary goal is to make reasonable predictions about unattested events *without* memorizing the details of the corpus. In an automaton with 10^6 states (quite feasible with today's technology) and 10^2 letters (the size of commonly used tagsets), we would have over 10^{14} free parameters, a huge number that could only lead to overfitting, were we to follow Suppes' dictum and restrict ourselves to precisely matching samples of size 10^{12}. Also, the high-level poverty of stimulus argument made in Miller and Chomsky (1963), that a childhood lasting 10^8 seconds is unlikely to be long enough for gathering sufficient data and computing 10^{14} parameters, renders the results of such brute force model fitting quite suspicious, a matter we shall return to in Section 3. But as long as we can be profligate with parameters, Conjecture 2 trivially holds, and not just for one-letter alphabets:

Theorem 3 For any p-language of any type $i \leq 3$ we can fit a PFSA to the first N terms of a distribution to any required precision.

Proof In fact, we can do more: given any p-language $f : T^* \to \mathbb{R}_0^+$ and any positive ϵ we can find a PFSA with rational probabilities such that the p-language $g : T^* \to \mathbb{R}_0^+$ generated by it satisfies $\sum_{\alpha \in T^*} |f(\alpha) - g(\alpha)| \leq \epsilon$ i.e. the total absolute error of the approximation is bound by ϵ. First, we arrange the strings of T^* lexicographically, and select an N such that the total probability mass assigned to strings $\{s_i | i > N\}$ is less than $\epsilon/2$ – this is always possible since f converges. Next, we approximate $f(s_0)$ by a rational within $\epsilon/4$, this will be $g(s_0)$, and similarly we approximate $f(s_1)$ within $\epsilon/8$, $f(s_2)$ within $\epsilon/16$ and so on, for a total error $\leq \epsilon$. Finally, we construct the PFSA to generate the first N approximate g values (the remaining values of f will be approximated by 0) by creating a separate chain automaton for each string s_i and using their probability $g(s_i)$, a rational number, as the transition weight for a λ-move from the initial state into the starting point of the i-th automaton. For a string $a_1 a_2 \ldots a_k$ the chain automaton moves from state i to state $i+1$ on emitting a_i with probability 1, so if only the last state is accepting it generates $a_1 a_2 \ldots a_k$ with probability 1. This, multiplied with the cost of the silent initial move, gives exactly the p-language g.□

Discussion Since in a corpus of size N the smallest probability distinction that can be empirically made is $1/N$, by picking $\epsilon = 1/2N$ in Theorem 3 we can construct a PFSA with all rational coefficients whose language will be statistically indistinguishable from the original p-language. Conjecture 1 express the belief that the Chomsky hierarchy will generalize smoothly from standard (unweighted) grammars and languages to the probabilistic (weighted) case. This was not an unreasonable expectation, especially in the light of what was known at the time about the Type 0 case. For Turing machines, an important reduction was presented in De Leeuw et al (1956), showing that a TM with access to a random number generator that produces 1s and 0s with some fixed probability p is equivalent to a standard TM without random components as long as p itself is computable – in other words, adding random operations to the deterministic Turing operations adds nothing to generative capacity. In light of this, Ellis' (1970) result came as something of a nasty surprise, and it was quite reasonable for Suppes to look for a fallback position such as Conjecture 2.

Any string of 0s and 1s can be uniquely mapped on a language over a one-letter alphabet: we set $a_i \in L$ if the i-th digit was 1 and $a_i \notin L$ if it was 0. If the string was generated by a random number generator with a fixed probability p for 1s, the density of the associated language, defined as $\lim_{n\to\infty} \frac{|\{a^i \in L : i < n\}|}{n}$ (see Eilenberg 1974, Kornai 1998) will be p. It is easy to program a Turing machine that outputs a string (or language) with no density, but it is impossible to program a TM that outputs (or accepts) a language with non-computable density. In other words, the key issue is not the randomness of the machinery, but rather the complexity of the real numbers that express the probabilities, and in this sense Suppes is entirely right in excluding the pathological cases. This is a step in the right direction, but as Example 1 shows, it is not enough: the key issue is that probabilistic grammars have a structure on their own right, and this structure is largely unrelated to the numerical values the probabilities may take.

At the bottom of the complexity hierarchy we find the rationals: every rational number between zero and 1 (and only these) can be the density of an unweighted language over a one-letter alphabet generated/accepted by some FSA. Adding weights $p_1, \ldots, p_k$ to a FS grammar or automaton accomplishes very little, since we will of necessity remain in the rational closure of these. However, real numbers are powerful carriers of information: all algorithms can be encoded in a single real number. When we add just one non-computable p to the weight structure, we have already stepped out of the TM domain as de Leeuw et al (1956) demonstrated.

At the next level of the hierarchy we find the algebraic numbers: these correspond to polynomials and thus, via generating functions, to CFGs and CFLs. While the relationship is not very visible over the one-letter alphabet, because CFLs have letter-equivalent FSLs, and one needs to consider the structure to see the difference, algebraic numbers (and only these) arise naturally at every stage of the analysis of unweighted CFGs – for the weighted case the same caveat about introducing arbitrary reals applies.

As for Suppes' objection to Ellis' construction, all algebraic numbers, and some transcendental ones, are computable. By Liouville's Theorem, algebraic numbers of degree n can only be approximated by rationals p/q to order q^{-n} – if a number can be approximated at a faster rate it must be transcendental. (Of course, it may still be transcendental even if it has no faster approximation.) Arguments based on transcendence can be thought of as convenient shorthand for arguments based on order of growth. The analogy is loose, and we shall not endeavor to make it more precise here, but Theorem 2 is a clear instance of replacing the heavy machinery of transcendence by a more pedestrian reckoning of growth (in our case, exponential decay).

3 Conclusions

Having acquired a more nuanced understanding of the role that probabilities play in p-grammars, let us finally turn to some of the computational implications and consider Hidden Markov Models (HMMs) which, in a theoretical sense, are just probabilistic finite state transducers between strings of hidden states and strings of observables. Given that HMMs in generation

mode are formally equivalent to PFSA, and the latter by Theorem 2 show exponential probability decay as the generated strings get longer, the expectation is to find the same probability decay in HMMs. Yet in practical applications, this is not quite what we find, and it is worth looking at the root of this phenomenon. Let us consider some rough estimates of the number of parameters in an n-gram POS tagger HMM. In the unigram case, the hidden states are simply the POS tags, so there are on the order of 10^2 to consider, and there are 10^4 transition parameters. In the general case there are 10^{2n} states and 10^{2n+2} nonzero transition parameters: Miller and Chomsky (1963) call attention to unbounded dependencies such as *The people who called and wanted to rent your house when you go away next year are from California* and suggest $n = 15$.

In a unigram model, there are only a few times more emission probabilities than words in its dictionary, since each word can have only a few POS tags: 10^6 is a reasonable estimate. In a bigram model there will be about 10^8 and in a trigram model 10^{10} emission values, and it is common practice to tie these together (i.e. to make emissions dependent only on the last member of a POS n-gram, as in TnT, see Brants 2000) in order to avoid overfitting. The typical nonzero transition probability (zeros are actually replaced by small numbers obtained from smoothing in practical applications) is in the range $10^{-1} - 10^{-4}$, while the typical emission probability can vary greatly, as much as 8-9 orders of magnitude, within a single (n-gram) state. Thus the probability differences between sentences are largely driven by the emissions, while the exponential decay comes from the transitions alone. In other words, in the process of fitting a model to an observed distribution, the straight PFSA model will of necessity have a large number of very low transition parameters, causing both overfitting and a very fast exponential decay, while a straight HMM (which is a PFSA but with parameters tied in a manner more appropriate to the task) will have higher transition probabilities (slower exponential decay) and much less of an overfitting problem.

In general, Theorems 2 and 3 don't stand in the way of training statistically reasonable language models, they just signal the difficulty, overfitting, that the naive empirical approach runs into. The answer is not to abandon fitting regular models, or to abandon the probabilistic framework entirely as suggested e.g. in Chomsky (1957), but to proceed with the appropriate parameter tying strategies.

Acknowledgements

The comments from anonymous MOL reviewers and from András Serény (CEU) resulted in significant improvements and are gratefully acknowledged here.

References

Thorsten Brants. 2000. TnT – A statistical part-of-speech tagger. In *Proc ANLP 2000*, pages 224–231.

Noam Chomsky and Marcel Paul Schützenberger. 1963. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 118–161. North-Holland, Amsterdam.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.

K. de Leeuw, E.F. Moore, C.E. Shannon, and N. Shapiro. 1956. Computability by probabilistic machines. In C.E. Shannon and J. McCarthy, editors, *Automata studies*, pages 185–212. Princeton University Press.

Samuel Eilenberg. 1974. *Automata, Languages, and Machines*, volume A. Academic Press, San Diego, California.

Clarence A. Ellis 1970. *Probabilistic languages and automata.* University of Illinois, Urbana Ph.D. Thesis

András Kornai. 1998. Quantitative comparison of languages. *Grammars*, pages 155–165.

András Kornai. 2007. *Mathematical linguistics*. Springer.

W.J. M. Levelt. 1974. *Formal Grammars in Linguistics and Psycholinguistics, Vol. II: Applications in Linguistics Theory*. Mouton, The Hague.

George A. Miller and Noam Chomsky. 1963. Finitary models of language users. In R. Duncan Luce, Robert R. Bush, and Eugene Galanter, editors, *Handbook of mathematical psychology*, pages 419–491. Wiley, New York.

Fernando Pereira. 2000. Formal grammar and information theory: Together again? *Philosophical Transactions of the Royal Society*, 358 (1769):1239–1253.

Patrick Suppes. 1970. Probabilistic grammars for natural languages. *Synthese*, 22:95–116.