JoLLI manuscript No.
(will be inserted by the editor)

Probabilistic Grammars and Languages

András Kornai

Received: date / Accepted: date

Abstract Using an asymptotic characterization of probabilistic finite state languages over a one-letter alphabet we construct a probabilistic language with regular support that cannot be generated by probabilistic CFGs. Since all probability values used in the example are rational, our work is immune to the criticism leveled by Suppes (1970) against the work of Ellis (1969) who first constructed probabilistic FSLs that admit no probabilistic FSGs. Some implications for probabilistic language modeling by HMMs are discussed.

Keywords probabilistic grammar · probabilistic language · PFSA · PCFG

1 Background

We tend to think of mathematical theorems as eternal, clearly demonstrating the conclusion from the premises. Here we will deal with the fate of a classic theorem of Ellis (1969), which serves as something of a cautionary tale: no matter how clear the mathematical evidence in favor of a particular view, people vested in an opposing viewpoint will find a way around it.

Ideas from statistics and probability theory have penetrated our scientific thinking to such an extent that

> shortly after 1930 it became virtually certain that at bottom our world is run at best by laws of chance. In epistemology [...] it is now a commonplace that much of our learning from experience, and much our foundations for knowledge, are to be represented by probabilistic models. (Hacking 1987)

Yet for the past fifty years, theoretical linguistics has largely steered clear of statistical argumentation. Chomsky (1957) presented the famous *colorless green ideas sleep furiously* argument purporting to show that the very notion of probability is linguistically meaningless, since *sleep ideas furiously green colorless* also has zero probability, yet the first sentence is grammatical while the second is not. Another widely read and cited paper from the formative period of generative grammar, Miller and Chomsky (1963), used the poverty of

Institute for Quantitative Social Science,
Harvard University
1737 Cambridge St, Cambridge MA 02138
E-mail: andras@kornai.com

Computer and Automation Research Institute,
Hungarian Academy of Sciences
13-17 Kende u, H-1111 Budapest, Hungary
http://kornai.com

stimulus argument to show that 'reasonable' Markov models cannot possibly be learned by children, who will not be exposed to more than 10^9 sentences (assuming one sentence per second for 24 hours a day for their first 20 years of life). Their notion of reasonableness required a Markovian parameter k large enough to capture grammatical phenomena such as number agreement in constructions like *The people/person who called and wanted to rent your house when you go away next year are/is from California*, where the grammatically connected elements are separated by 14 or more units.

During the neogrammarian and structuralist periods, in keeping with the general tenor of scientific research, linguistic arguments based on statistical notions, ranging from the simple counting of cases supporting or escaping some rule to the sophisticated criteria for segmentation proposed by Bloch,[1] were treated as a standard tool of inquiry. With the advent of generative grammar, probability fell in grave disfavor. In the flurry of research activity that brought us transformational grammars, time-varying grammars, matrix grammars, programmed grammars, ordered grammars, scattered context grammars, indexed grammars, and many other algebraic string manipulation systems, probabilistic grammars barely made a blip on the radar screen. To the limited extent they were investigated, probability was generally treated as an afterthought, giving primacy to the algebraic characterization of the system. What we have at this point are *weighted* grammars, where the weights can be taken from any number field, and not at all required to be probabilities: examples include integer weights, counting the multiplicity of derivations (degrees of ambiguity) and acceptance level weights (degrees of grammaticalness). The mathematical underpinnings of using weights in the context-free case are published in Chomsky and Schützenberger (1963), but the issue of degrees of grammaticalness is already explored in Chomsky (1961).[2]

Into this situation walked Skip Ellis, the first ever African American to earn a PhD in computer science. He proved 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. The conclusion seems quite clear: the variety of probabilistic structures cannot be reduced to the variety of algebraic structures, *there are more things in probabilistic heaven and earth than are dreamt of in your grammatical philosophy*. This conclusion, however, was not one that linguists or philosophers of language were ready to hear at the time. Scientometry is a crude tool, but in this case the facts are rather striking: according to Google Scholar the dissertation where the proof appeared gathered a total of 28 citations (one from this author), while the debunking paper, Suppes (1970), is cited 70 times.

Ellis' proof (which we strengthen in Section 2 from PFSGs to PCFGs, Theorem 1) is highly algebraic, and gives no indication *why* such p-languages exist. Suppes (1970) argued that Ellis' result is irrelevant and his construction is just an artefact of using irrational probability values, since the probabilities that matter all go back to frequency counts i.e. rational numbers. What is at stake here is the nice, well-groomed picture of p-languages, which should neatly fit in the Chomsky hierarchy. The goal was to treat the algebraic model of string-manipulation as fundamental and the statistical issues as secondary. One way to

[1] In contemporary terms, Bloch used local perplexity maxima to find morpheme boundaries, see Harris (1951)

[2] The earlier publication date is not necessarily indicative of an earlier stage in Chomsky's thinking. At this point, much of Chomsky's own work is devoted to further exploration of material intended for his PhD thesis, written in 1955-56. Not all chapters of this ms. were widely read, since the thesis, as published by University Microfilms, omitted several, and so did the 1975 book version (Chomsky 1975). In 2007 the full ms. was made available as pdf, http://alpha-leonis.lids.mit.edu/chomsky. Fully tracing the provenance of these ideas is a task beyond the limits of this paper.

achieve this is by treating statistical issues by a mere retrofitting of the grammar rules with application probabilities: a type i probabilistic grammar (or p-grammar for short) is defined as a type i grammar where all productions have some nonzero probability of application. If the sum of probabilities assigned to productions sharing the same left hand side is 1, we speak of *normalized* p-grammars – these formalize the concept that derivations proceed in some random order as dictated by the probabilities assigned to the rules. Once the rules are viewed as probabilistic entities, the probability of a *derivation* is the product of the probabilities of the rules that appeared in it (with multiplicity), and the probability of a *string* is just the sum of the probabilities of its derivations. Remarkably, the concept of rules as probabilistic entities embodied in this retrofit is not at all close to the concept of probabilistic rule application that the the minority of linguists who cared about such matters began to explore, *variable rules:* in p-grammars the rule probabilities are static entities, fixed once and for all, quite independent of the contextual factors that sociolinguists took to be the central point of inquiry.

It was clear from the outset that there are problems with this nice picture: as is well known (see e.g. Levelt 1974), 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. But these problems can be easily paved over, e.g. by assigning the missing weight to the empty string or by renormalizing the productions. Also, the issue that the probabilities do not add to one, or will add to one in every annulus (set of strings with the same length) was seen as less than worrisome – the fine distinction between a Markovian *language* and a Markovian *process* was really not for linguists to worry about. Now if Ellis proves the existence of p-languages that apparently tear apart the probabilistic generalization of the leading theoretical construct of the period, the Chomsky hierarchy, so much the worse for these p-languages. Altogether, since 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 retrofit makes the proof quite 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 nonzero 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 at most n steps by leftmost derivation 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 renormalize to absorb the difference). It is in the other direction that Ellis (1969), found a stumbling block: even over a one-letter alphabet, and already among type 3 languages, he presented a p-language L_3 for which no finite state p-grammar (or p-acceptor) will ever work. In Section 3 we will 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).

As the simplest case, studying the issue over a one-letter alphabet $T = \{a\}$, has methodological priority: if counterexamples can be found in this limited setup, they are trivial to extend to alphabets with more letter, while the converse is not true. 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.

2 The Suppes/Levelt conjecture

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 right-linear grammars generate strings with probabilities that sum to 1. However, the relationship between right-linear grammars and finite automata is more complex in the probabilistic case, and we will settle on a definition of probabilistic finite state automata (PFSA) that is different from the classic definition of probabilistic automata given in Rabin (1963). The issue is that in the typical case, Rabin-style automata define probabilistic *processes*, but do not define probabilistic languages. To see this, consider Rabin's first example, a machine with two states 0 (the start state) and 1 (the only accepting state), over a two-letter alphabet $\{0,1\}$, with transitions for symbol i being defined as a loop (probability 1) over state i, and as equiprobable between states i and $1-i$ out of state $1-i$. This automaton assigns the string 1 probability 0.5, the string 11 probability 0.75, and in general the string 1^n probability $1-2^{-n}$. In other words, the values assigned by the machine do not converge. We obtain a full probability distribution for every annulus L_n composed of exactly n-letter strings, but not for the language as a whole.

In a probabilistic language, we could always extend p to assign 0 to all strings not in L, and it is generally sufficient to speak of functions $f : T^* \rightarrow \mathbb{R}_0^+$ for which $P = \sum_{\alpha \in T^*} f(\alpha)$ is convergent. The exact value of P is not a central concern: if $P < 1$ we can always pad it out (e.g. by assigning the remainder to the empty string) and if $P > 1$ we can divide all values by P. We identify the p-language L as the *support* of f i.e. the set of strings α for which $f(\alpha) > 0$. (In Rabin's terminology, we assume the *cut-point* to be 0.) In particular, a probabilistic language always induces a *length distribution* given by $p(n) = \sum_{\alpha \in L_n} f(\alpha)$. Since actual corpora have a length distribution with a finite mean (e.g. the average sentence in the Brown corpus is less than 25 tokens long), it only makes sense to treat corpora as samples from probabilistic languages, as opposed to probabilistic processes. Technically, the issue amounts to the existence (or lack thereof) of a *sink state* the automaton gets into when presented with an end marker. We return to the matter shortly when we formally define PFSA, but note here that the trivial machine, with one (accepting) state and all input looping back with probabilities summing to 1, will not be a PFSA as it lacks a sink.

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 $s_1, s_2, s_3, \ldots, s_{n+1}$ cannot all be obtained from the CFG in question. $\square$

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.

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. That this was generally viewed as highly desirable is clear from Levelt (1974:44), who takes the extraordinary step of listing this as a theorem. Even though the surrounding text makes the conjectural status quite clear, this is truly a momentous decision: it would be hard to imagine a number theory textbook presenting the 'Goldbach Theorem' or the 'Riemann Theorem' in boldface, leaving the reader with the impression that essentially all the work is done, no further research is needed.

Conjecture 2 (Suppes/Levelt) p-languages with type i support are statistically indistinguishable from languages generated by type i p-grammars.

Here we will investigate whether this 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 4.

3 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 $i > 0$ – this has the technical advantage that instead of a single designated initial state we can now have an initial probability distribution. We also 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.

Discussion Consider the trivial single-state Rabin automaton over a one-letter alphabet $\{a\}$: if we wish to write a right-linear grammar for this case we notice that the rule $S \rightarrow aS$ is insufficient (there is no way to eliminate the S from a sentential form) so we must add a rule $S \rightarrow \lambda$. But once we do this, the probability p of the rule $S \rightarrow aS$ must be set to strictly less than 1, since in p-grammars all rules, including the rule that eliminates the nonterminal, must have nonzero probability, and the total probability of rules rewriting S must be kept at 1. This restores convergence, both in that the total probability assigned will now be 1 and in that the mean string length is now finite, $p/(1-p)$.

In a PFSA 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 $p : \{a\}^* \rightarrow \mathbb{R}^+$ 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 (2008) 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 kth 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 $N > 10^{10}$ (in the world of machine learning, there is no poverty of stimulus in the sense urged by Miller and Chomsky) 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. 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}. The poverty of stimulus argument still holds in that a childhood lasting 10^8 seconds is unlikely to be long enough for gathering sufficient data and computing 10^{14} parameters, 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 $\sum_{\alpha \in T^*} |f(\alpha) - g(\alpha)| \leq \varepsilon$ 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 $\varepsilon/2$ – this is always possible since f converges. Next, we approximate $f(s_0)$ by a rational within $\varepsilon/4$, this will be $g(s_0)$, and similarly we approximate $f(s_1)$ within $\varepsilon/8$, $f(s_2)$ within $\varepsilon/16$ and so on, for a total error $\leq \varepsilon$. 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 Theorem 3 makes the Suppes/Levelt conjecture come out literally true, but destroys the entire intellectual agenda that led to it. Once we invoke the notion of statistical indistinguishability, there is *no* correspondence between type i p-grammars and type i p-languages: all that remains is that every p-language can be approximated by a PFSA to any degree, just as every number can be approximated by a rational to any degree. In a corpus of size N the smallest probability distinction that can be empirically made is $1/N$. By picking $\varepsilon = 1/2N$ in Theorem 3 we can construct a PFSA with all rational coefficients whose language will be statistically indistinguishable from the original p-language.

4 Conclusions

Conjecture 1 expressed the belief that the Chomsky hierarchy will generalize smoothly from standard (unweighted) grammars and languages to the probabilistic (weighted) case. When Ellis showed that this is untrue, Suppes sought out a fallback position, Conjecture 2, expressing the same belief. This was not an unreasonable move, especially in 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. There is nothing wrong with the strategy Suppes tried, excluding the pathological cases. This could have been a step in the right direction, but as Example 1 shows, it was not: the key issue is that p-grammars have a structure on their own right, and this structure is largely unrelated to the numerical values the probabilities may take. Whether these numbers are rational, algebraic, or transcendental, matters but little for the purpose of statistical language modeling, since every number of the more complex transcendental type can be replaced by a simple (rational) number with any desired precision.

What of the road not taken for so long? In linguistics, the brilliant technical work of Chomsky and Schützenberger (1963) served more as a boundary marker than as a starting point for further work – few people felt comfortable going in a direction Chomsky himself declared useless. In mathematics, the work was not only preserved but significantly enlarged by Schützenberger, Berstel, and their students. Yet in spite of a lively 'French School' that continued to produce excellent work (of which we single out Schützenberger 1976, Reutenauer and Schützenberger 1991, Berstel 1973, 1979, 1988), it wasn't until the groundbreaking work of Mohri (1994), himself a student of Berstel, that this line of research caught the interest of linguists again. The original weighted FST library that Mohri and his colleagues built at AT&T (see Mohri, Pereira and Riley 1996, Kornai 1996) had strong proprietary restrictions, but an open source version is now widely used. In some respects, the story of weighted finite state automata and transducers is eerily reminiscent of the unweighted case, of which we wrote the following:

> To understand some of the main trends in finite state NLP it is worth looking back at the origins of the field. Though neither Mealy (1955) nor Kleene (1956) had NL applications in mind, finite state methods were applied in this domain as early as 1958 (see Joshi and Hopely 1999). In the early sixties, however, finite state models were soon submerged in a flood of transformational models. It is hard to speculate about such matters, but it is quite conceivable that the finite state approach to NLP would have lost all credibility, were it not for the extraordinary impact of Thompson (1968) and the `grep` family of unix tools. While theoretical linguists accepted the arguments put forth in Miller and Chomsky (1963) at face value, from

> the seventies it became part of the received computer science wisdom that if you want to do something with text you need to build finite automata. By making his implementation of `regexp(3)` freely redistributable, Spencer (1986) transmitted this wisdom to the free software movement. Given the dominant position of finite state technologies in topic search, in retrospect it is hard to understand why mainstream syntactic theory continued to shun finite state methods throughout the seventies and eighties, but in fact these methods reappeared on the scene through a back door left open by the context sensitive rule systems of phonology. Only two years after the seminal Sound Pattern of English (Chomsky and Halle 1968), Johnson (1970) demonstrated that the context sensitive machinery of SPE can be replaced by a much simpler one, based on finite state transducers (FSTs), and independently the same conclusion was reached by Kaplan and Kay, whose work remained an underground classic until it was finally published in 1994. (Kornai 1999).

The wake-up call delivered, however inadvertently, by Ken Thompson, a computer scientist with little discernible interest in language, was rather mild compared to the harsher wake-up call delivered by two electrical engineers, Fred Jelinek and Jim Baker, whose speech recognition system, based on Hidden Markov Models, massively outperformed the rule-driven systems popular at the time (Woods et al 1976). By the time Brown at al (1990) established that not just phonetics and phonology, but also syntax and semantics (and in particular the Holy Grail of half a century's work, machine translation) can benefit from statistical ideas, few people entering the field thought that the received wisdom, still a part of the standard linguistics curriculum, was valid.

But what of the objections to statistics that held back this line of research for three decades? The first one, as emphasized in Pereira (2000), is simply false: the fact that *colorless green ideas sleep furiously* and *sleep ideas furiously green colorless* both have zero *frequency* by no means implies that they have zero *probability,* or even the same nonzero probability. As for the poverty of stimulus argument, this certainly remains valid, indeed, this may be the only form of the argument that is robust enough to stand up against the objections of Pullum and Scholz (2002). However, from an engineering standpoint there is nothing particularly worrisome: all that is required is to create models that have fewer parameters, and much of contemporary linguistic theory aims exactly at this goal, primarily by exploring various parameter tying strategies. To see what is involved here, 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 n-gram case there are 10^{2n} states and 10^{2n+2} nonzero transition parameters: Miller and Chomsky (1963) are quite right in noting that 10^{32} parameters are impossible to train (or to acquire as a human).

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 in Theorem 2 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 do not 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 was suggested e.g. in Chomsky (1957), but to proceed with the appropriate parameter tying strategies.

Acknowledgements The comments from András Serény (Central European University), Larry Moss (Indiana University), and from the anonymous MOL reviewers, resulted in significant improvements and are gratefully acknowledged here. We also acknowledge the unsung hero of this piece, Skip Ellis, who had the best possible revenge, becoming a highly cited and very influential computer scientist (his top three papers have over 3,000 citations).

References

[Berstel1973] Berstel, Jean. 1973. Sur la densité asymptotique de langages formels. In M. Nivat, editor, *Automata, Languages, and Programming*. North-Holland, pages 345–358.

[Berstel1979] Berstel, Jean. 1979. *Transductions and Context-Free Languages*. Teubner Studienbücher, Stuttgart, Germany.

[Berstel1988] Berstel, Jean. 1988. Finite automata and rational languages, an introduction. In *Lecture Notes in Computer Science*, volume 386. Springer.

[Brants2000] Brants, Thorsten. 2000. Tnt – A statistical part-of-speech tagger. In *Proc ANLP 2000*. pages 224–231.

[Brown et al.1990] Brown, Peter, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85.

[Chomsky1957] Chomsky, Noam. 1957. *Syntactic Structures*. Mouton, The Hague.

[Chomsky1975] Chomsky, Noam. 1975. *The logical structure of linguistic theory*. Springer Verlag.

[Chomsky and Halle1968] Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. Harper and Row, New York.

[Chomsky and Schützenberger1963] Chomsky, Noam and Marcel Paul Schützenberger. 1963. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*. North-Holland, Amsterdam, pages 118–161.

[de Leeuw et al.1956] de Leeuw, Karel, Edward F. Moore, Claude E. Shannon, and N. Shapiro. 1956. Computability by probabilistic machines. In C.E. Shannon and J. McCarthy, editors, *Automata studies*. Princeton University Press, pages 185–212.

[Ellis1969] Ellis, Clarence A. 1969. *Probabilistic Languages and Automata*. PhD thesis, University of Illinois, Urbana.

[Hacking1987] Hacking, Ian. 1987. Was there a probabilistic revolution 1800-1930? In L. Krüger, L. Daston, M.Heidelberger, G. Gigerenzer, and S. Morgan, editors, *The Probabilistic Revolution*. MIT Press, Cambridge MA.

[Harris1951] Harris, Zellig. 1951. *Methods in Structural Linguistics*. University of Chicago Press.

[Johnson1970] Johnson, Ch. Douglas. 1970. *Formal aspects of phonological representation*. PhD thesis, UC Berkeley.

[Joshi and Hopely1999] Joshi, Aravind K. and Philip Hopely. 1999. A parser from antiquity. In András Kornai, editor, *Extended Finite State Models of Language*. Cambridge University Press, pages 6–15.

[Kaplan and Kay1994] Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20:331–378.

[Kleene1956] Kleene, Stephen C. 1956. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, pages 3–41.

[Kornai1996] Kornai, András. 1996. Comments on Mohri, Pereira and Riley. In András Kornai, editor, *Proceedings of the W1 workshop of the 12th European Conference on Artificial Intelligence*, pages 26–27, Budapest. John von Neumann Society for Computer Science.

[Kornai1999] Kornai, András, editor. 1999. *Extended Finite State Models of Language*. Cambridge University Press.

[Kornai2008] Kornai, András. 2008. *Mathematical Linguistics*. Springer Verlag.

[Levelt1974] Levelt, Willem J.M. 1974. *Formal Grammars in Linguistics and Psycholinguistics*, volume 1–3. Mouton, The Hague.

[Mealy1955] Mealy, George H. 1955. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34:1045–1079.

[Miller and Chomsky1963] Miller, George A. and Noam Chomsky. 1963. Finitary models of language users. In R.D. Luce, R.R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*. Wiley, pages 419–491.

[Mohri1994] Mohri, Mehryar. 1994. Minimisation of sequential transducers. In *Lecture Notes in Computer Science, Proceedings of the Conference on Computational Pattern Matching 1994*. Springer.

[Mohri, Pereira, and Riley1996] Mohri, Mehryar, Fernando C.N. Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In András Kornai, editor, *Proceedings of the ECAI-96 Workshop on Extended Finite State Models of Language*, pages 46–50, Budapest. John von Neumann Computer Society.

[Pereira2000] Fernando Pereira. 2000. Formal grammar and information theory: Together again? *Philosophical Transactions of the Royal Society*, 358 (1769):1239–1253.

[Pullum and Scholz2002] Pullum, Geoffrey K. and Barbara C. Scholz. 2002. Empirical assessment of stimulus poverty arguments. *The Linguistic Review*, 19:9–50.

[Rabin1963] Rabin, Michael O. 1963. Probabilistic automata. *Information and Control*, 6:230–245.

[Reutenauer1991] Reutenauer, Christophe. 1991. Subsequential functions: characterizations, minimization examples. In *Proceedings of Internantional Meeting of Young Computer Scientists, Lecture Notes in Computer Science*.

[Schützenberger1976] Schützenberger, Marcel Paul. 1976. Sur les relations rationnelles entre monoides libres. *Theoretical Computer Science*, 3:243–259.

[Spencer1986] Spencer, Henry. 1986. `regexp(3)`. Posted on `mod.sources` 3(89).

[Suppes1970] Suppes, Patrick. 1970. Probabilistic grammars for natural languages. *Synthese*, 22:95–116.

[Thompson1968] Thompson, K. 1968. Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422.

[Woods et al.1976] Woods, William A., Madelaine Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, John Makhoul, Bonnie Nash-Webber, Richard Schwartz, J. Wolf, and Victor Zue. 1976. Speech understanding systems: final technical progress report.