4TM

10 : In a Phase 2 TM, we would be Modeling h's h' (194.30 : pp16–19 of my IDS IA Report)
01 often using ideas found by PPM's modifns, & 198.30 (166.11–.40)
Actually, in Phase 1 we would be doing .00–.01 as well.

0 : 194.17 : T. Monte-Carlo search does not at all work well for inv probs w. only 1 soln.
Hvr, how it works for normal OZ probs is unclear! It is "sort of" used in GA, where it clearly is much better than pure Monte Carlo!
How GA works: use McCarlo to get a subset of cands of > average G. Using mut/cross, derive a new PD that fits that subset of cands & loop to α.
Actually, the only difference (w.o. Lsrch) of my method v.s. conventional GA, is that I use ssz > 2 a' priority & use much better models of the GPD.
T. magnitude (& other effects) of Lsrch relative to McCarlo, is not clear!
M Carlo has more variance, so it would seem more able to find "crazy far out solns" — but using better GPD models may be better. In any case, the "de facto" effect of Lsrch is not clear. — degree of goodness/badness of

0 : On Hyer level lrng: ~TM$_2$! Say we are in Phase 1 — we start thru TSQ:
Th. regy needed is not findable by PPM. What we do is Go to hyer level: We use TM1 to use a complete (univ.) lang to find regys needed & incl. TSQ. This hyer level can have a choice betw. several update methods. One is PPM, another improved PPM ... another use of ~ PSGD. or others! Go over this More carefully: more exact statement
26 of what to do — A Picture ("in English") of what I expect Phase 1 ... ("4") to do.
① Simple case: TSQ of QA problems: TSQ is "short" in the sense of not many new concs. introduced. Also solns. have short solns in ref. lang. GPD = ref p.d. .... Ordering of
probs in TSQ is irrelevant to speed of soln. Essentially, we have this list of rules & their
30 pc's & we do an Lsrch to solve all probs in TSQ. There is no modifn of t. GPD !.
no "hyer level lrng". Static GPD: Perhaps based on props of Tokens only,
32 ② As above but GPD is updated, say using PPM. PPM may need large ssz's — so one way → 205.00
of simulating this is to repeat problems. W. each QA problem we give TM a number that tells how many times it was repeated. This gives Signif Modifn of PPM update.
We are still limited in poss. TSQ's that t. system could lrn to understand/predict.
—— Tho it may be that very many regys are expressable as simple concats of items.
Tho once a v.g. set of items is discovered, & one has probys for various concats of them, (20000

p30 ①

.20 : 197.40 (on p33) : On OOPS lang: OOPS: p33 lookat defn of FAC & FAC2.

③ defn (1, 1, Fac, up c1 ex rt0 del up dec FAC mul ret)

- no. input args; no. outputs; Name of funct. (for user)
- up: p30 ⑤ dups top of ds
- c1 (integer)
- ex: p30 ⑤ exchanges 2 top elements of ds
- rt0: p31 ③ Return if arg ≤ 0
- del (p30 ③): dec dp = search ptr of ds
- up: documents search ptr of ds
- dec: decrements ds; returns "x-1"
- FAC mul: multiplies 2 TOS objects FAC(3 ones) Another argt of FAC
- ret: return

.30 Q: It would seem that OOPS' main memory of useful seqs consists of the first would be via Definitions — sol: GPD the definitions being made while pgms are created (not during update of Guiding P.D." (= GPD))

AZ would operate in similar way. The alternative is to have defns discovered in GPD update — or ≈ equivt. By using PPM in GPD update. We could, of course, have both.

The motivation of TM to make defns during trial cand generation, would seem to be small. Usually defining a sequence is a waste of pc & cc.

Hvr, the "normal" way of Defining Recursive functions in Lisp, OOPS (& perhaps AZ) does involve ability to define non-recursive functions. On the otherhand, we'd like recursion to be "Heuristically enabled" by a mechanism like 197.09 – .15. ← which would seem to be in GPD update.

→ [Closer to TOwer of Hanoi!] Say we have as seq. of n problem defns,

If φ_1 solves prob$_1$ & φ_2 solves prob$_2$, then If $\varphi_2 = f(\varphi_1, 2)$ & F is "simple", then $f(\varphi_2, 3)$ is a good cand. for problem 3.

.0 Hvr, it is perhaps possl. to have some good prob. solving w.o. the usual heuristics! i.e. all of these are built into GPD update (like PPM, psedizy, ect)

So: What are the problems? 1) Why would TM define sub functions in Cand generation mode? ← PC was too low

2) A fortiori for Recursively defined functs. (a fortiori)

3) PPM seems to elim need of defns (to some extent) — But not completely & it doesn't enable recursion.

.27 Two Disadvantages of a Defn: Replacing thing defined by a Token, one loses all info about the internal Structure of the thing defined — Also one choses a particular parsing ‖ PPM avoids both of these. ← Reason a defn is applied. A particular parsing also abandons possy of alternate structures

.29 The disadvantage of PPM (for defns), it does not Ext'n by appy. to concat of 2 defined ngms. ——
.30 Some time Back, I worked out a method to detect ngms. in PPM (166.11) — It had to do w. a 2 dim. Representation of the Corpus data. Something like Z(i, j) = Case count for contexts ending at i & of length j.

It may be possl. to detect concats of pairs this way: high cc, hvr., triplets also but even more cc.

The cc isn't exactly n^2 for pairs & n^3 for triplets, because we will only consider relatively short ngms/contexts)

In spite of the disadvantages of .27, definitions enable compression of corpus, so higher order defns are more easily seen. They are kind of SUMAC!, so to regression has to occasionally backtrack by undefining certain sequences (reparsing). } Main advantage

SO: Prob'ly a good strat. (Obvious by humans), is to tentatively make definitions — use them when it's rather clear that they are appropriate & reparse corpus w. the new defns. How to assign pc's to the defns is

.39 unclear, due to possys of many ‖ parses.

10:196.40 : On defns of Recursive functs: If t. lang has Do loops (I don't know if "while or until" are needed) it can do all prim. rec. functs: — which is probly adequate.

Hvr, recursive functs as such (via recursive defns) are (probly) needed to get hy pc for certain functs.

— So I'd like a way to define Rcsn, — then have t. compiler create a "Do loop" for implementation.

Kleene's Va (p209) ≡ my page 196½ : $\varphi(0)=q$; $\varphi(y+1)=\chi(y,\varphi(y))$

is a simpler way to define most prim rec. functs.

{ I had thought $\varphi(a)=q$ $\varphi(f(y)) = \chi(y, \varphi(y))$ Kleene uses $f(y)\equiv y+1$.

.09 One (common) practical way to get recursive solns: say we notice that a & b are both acceptable objects ; that b bears a simple relation, r, to a (i.e. $b=r(a)$.) We then hypothesize that t. set $r^{(n)}(a)$ will be "acceptable".

Unfortly, recognizing that $b=r(a)$ may be unlikely, since b & a may not be in proper form — for this reason, it is best to keep several solns. to each problem in memory. For both O Z & Inv. probs, one should "over search" for solns, if at all possl. This makes it more likely that a soln b $b=r(a)$ could be found.

Whatever means (formalisms) are used to define recursive functions, I suspect that they will usually be of very low pc a/o by CJS — (Unless found by .09–.15 ····which is a different kind of heuristic situation) I'm thinking now of simply generating various cands in "Guiding Pd" order — Recursively defined cands will usually have much longer dcrns than non-recursively defined cands. This may also be true of "Do loop" defined functions. So it may well be that recursive "Do loop" defd functions will not be found until we get around to large CJS problems. — But even for humans, such solutions of this sort of very large CJS.

Probably best way of dealing w. recursion is via "English" {194.24–.27; 196.10–.13, and ff.}

If I think that recursion is a good solution cand, then dcrb. t. recursion in English. Whatever way we decide to implement t. recursion, T. English statement of it should guide us in evaluating its pc.

Re: Recursion in OOPS! OOPS took an enormous amt of time to do Tow of Han w. n = 10 or 20 or 30 — In Lsrch, it must have tried other much more improbable pgms for shorter times .... See the 'Scans' — Did I really do "Lsrch"?

Hvr, Even if we stick to English, we do have to have at least 2 non-human languages: One for — This lang is part of t. "problem dcrn" — dcrbg t. problem (which can be close to English), One for dcrbg t. solution. — T. choice of this second can be impt. We can try out various langs to see which seem to work best. Take a look at t. OOPS language. Just how Recursion was implemented. Was it at all efficient? It may have been a bad problem! — i.e. t. Tower of Hanoi was bad — (Not bad for small n, hvr.) Even w. correct soln, it took enormous amt of time to solve. (for large n ··· for small n, not so large).

→198.00

amine this in more detail, let us write the pair of equations

$$\begin{cases} \varphi(0) = q, \\ \varphi(y') = \chi(y, \varphi(y)), \end{cases}$$

ss the definition of a function $\varphi(y)$ by induction on y, where q n natural number, and $\chi(y, z)$ is a given number-theoretic function ariables.

for example, the value $\varphi(4)$ is determined thus. To generate 4, rate successively 0, 1, 2, 3, 4. By the first equation, the value l be the given number q; then by the second equation, the value ll be $\chi(0, \varphi(0))$, i.e. (using the value $\varphi(0)$ already found) $\chi(0, q)$, ince $\chi(y, z)$ is a given function) is a given number; again the value l be $\chi(1, \varphi(1))$, i.e. $\chi(1, \chi(0, q))$; the value $\varphi(3)$ shall be $\chi(2, \varphi(2))$, $\chi(1, \chi(0, q)))$; and finally the value $\varphi(4)$ shall be $\chi(3, \varphi(3))$, i.e. $\chi(1, \chi(0, q))))$.

we have a process by which, to each natural number y, on the the generation of y in the natural number sequence, a corre- number $\varphi(y)$ is determined. Since a number $\varphi(y)$ is thus associat- , for each y, a particular number-theoretic function φ is defined se numbers $\varphi(y)$ as its respective values.

nction φ satisfies the equations (1), when (1) are considered as l equations in an unknown function φ, since every particular comprised in (1) (namely, $\varphi(0)=q$, $\varphi(0')=\chi(0, \varphi(0))$, $\varphi(1')=$, ...) is satisfied in the course of selecting the successive num- , $\varphi(1)$, $\varphi(2)$, ... Also this φ is the only function satisfying ictional equations, since the process by which we determined the e numbers $\varphi(0)$, $\varphi(1)$, $\varphi(2)$, ... from the equations (1) can be ed as showing that any function φ satisfying the equations must values selected.

er definitions by induction, the function φ defined depends on l variables $x_2, \ldots, x_n$, called *parameters*, which have fixed values ut the induction on y.

LE 1. Consider intuitively the equations

$$\begin{cases} a+0 = a, \\ a+b' = (a+b)', \end{cases}$$

encountered in the formal symbolism as Axioms 18 and 19. ine the function $a+b$ by induction on b, with a as parameter, a previously known function. Then the equations

$$\begin{cases} a \cdot 0 = 0, \\ a \cdot b' = (a \cdot b) + a \end{cases}$$

KLEENE : Meta mathematics

define $a \cdot b$ by induction on b, with $a+b$ as a known function; and

$$\begin{cases} a^0 = 1, \\ a^{b'} = a^b \cdot a \end{cases}$$

define a^b by induction on b, with $a \cdot b$ as a known function.

An example of a definition of a predicate by induction will be given later (Example 2 § 45).

What number-theoretic functions are definable by induction? To make this question precise, we must specify what functions are to be taken as known initially, and what operations, including what forms of definition by induction, are to be allowed in defining further functions.

We shall now select the specifications with a view to obtaining functions definable by induction in an elementary manner. These functions will be called 'primitive recursive'.

Each of the following equations and systems of equations (I)—(V) defines a number-theoretic function φ, when n and m are positive integers, i is an integer such that $1 \leq i \leq n$, q is a natural number, and ψ, χ_1, ... χ_m, χ are given number-theoretic functions of the indicated numbers of variables.

(I) $\varphi(x) = x'$.

(II) $\varphi(x_1, \ldots, x_n) = q$.

(III) $\varphi(x_1, \ldots, x_n) = x_i$.

(IV) $\varphi(x_1, \ldots, x_n) = \psi(\chi_1(x_1, \ldots, x_n), \ldots, \chi_m(x_1, \ldots, x_n))$.

(Va) $$\begin{cases} \varphi(0) = q, \\ \varphi(y') = \chi(y, \varphi(y)). \end{cases}$$

(Vb) $$\begin{cases} \varphi(0, x_2, \ldots, x_n) = \psi(x_2, \ldots, x_n), \\ \varphi(y', x_2, \ldots, x_n) = \chi(y, \varphi(y, x_2, \ldots, x_n), x_2, \ldots, x_n). \end{cases}$$

((Va) constitutes the case of (V) for $n = 1$, and (Vb) for $n > 1$.)

A function is *primitive recursive*, if it is definable by a series of applications of these five operations of definition.

This definition can be given in more detail, analogously to the definition of provable formula for the formal system (§ 19), say using the second version, as follows.

We refer to the above equations and equation pairs (I)—(V) as schemata. They are analogous to the postulates, with (I)—(III) in the role of axiom schemata (or more strictly, (I) to a particular axiom), and (IV) and (V) in the role of rules of inference.

A function φ is called an *initial function*, if φ satisfies Equation (I),

8/11/04
4TM

: Here it may be possl. to confine ourselves to Prim. rec. functs & avoid that recursive defn problem!
— See 193.10 on Pators (in Kleene)

I vaguely remember McCarthy defining $\{$ Fact(X) = 1 : X=1 else
Fact(X) = X Fact(X−1) $\}$

Use a If statement: Def Fact(X): If X=1 Then 1 else X * Fact(X−1).

This would do for simple rec. functs — more complex functions like Ackermann — maybe more diff.
In particular ... would one be able to "unroll them" (i.e. not use a stack?).

0:194.27 : This seems like a very impt. idea! I should keep dev. lang. in "English" as large as possbl, because it will enable me to get larger sample of kinds of abss. That it will need so in order in t. (Non-English) Reference (lang/ourc). Also use several English solns in ||, since I want
13 to be able to get as much "Milage" from t Corpus as poss). (Unclear as to how to get what I want from || solns). → 20

SN On NHTM: For real output, a distribution can be characterized by a μ & σ^2. For more exactness give more "Moments" or other params of t. D.F. Or, for Multimodal d.f. — have several μ, σ^2.
For discrete d.f's — One common approach: A "center of cluster" w. a rule telling how far other choices are from t. Center. For more complex d.f.'s — have > 1 cluster, possibly diffnt ways to express "difrnce from center" —

0 (·13) One way, When one has a "not such a hyp." soln. — one retains it because it might be useful for "backtracking" later.

.5: 193.11 : **On Prim. Rec. Funcs.** : P119 of Kleene pretty well clarifies Prim. recr funcs. — They are definable by simple functional operations. T. most complex involves a Do loop in which t. upper limit is defined by p.r. funct. (→ Copy is now by ! — 196½)
I don't know if Do "until" loops can be used to define p.r. funct. I suspect that they can but I haven't yet proved it.
We can define rational nos. as integer pairs.
P119 of Kleene enables us to define t funct for all integers ≥ 0 (="natural nos").

0: 193.03 Then Def → α BR β just defines completed branches, not "functions" (ordinarily).
In branch generation "BR" 15 a NT in PSG generated PFMs. In generation of t. branch in α BR β, if any Branch occurs, we can write "X_i" for it —
T. compiler regards this as an arg. & keeps track of t. "strip" of t. fondun being generated. If $i = 3$, say, t. next arg will have w. = pc, t. substripes 1 thru 4
We may want to modify this d.f., how say $i = 4$ w. pc = .5 $i = 1, 2, 3$ w. pc's $\frac{1}{6}$.
This looks pretty much like t. AZ language devtd in 1D&IM Report.

8/9 [illegible]

4TM G5 RE4 3363 ← Phone Bach ~ 7pm

On Minsky's "Chapter 6":

Progress in Statistical learning: Theoretical, practical.

1) PPM & its improvements: ~ better than 2 bits/symbol approaches 1.3 bits/symbol of Humans. ~4 bits for LZ; ~ 2.25 bits for Bzip2 — < 2 for improvements. cc. if parts of speech given → 1.3 for human

(also some techniques seem promising for getting much closer.)

2) Fogel's checker player: feedback from win or lose only using ANN and hill climbing. Feedback from win/lose only! A neat way to deal w. "credit assignment problem".

3) Koza's GP using Lisp pgms: Designing of Op amps, Invention of Patented Spectrum filters: Now common types of filters. — Also new patents

4) Mach Translation — Statistical { Lang understanding as a kind of MT problem .... Common Human lang + in Formal lang.

5) Recent improvements in GA sorts more like regular statistics — No populations — Passing of Leach ... not yet used. "Cracking" of PSG discovery problem? [Bayesian Modern GA!]

6)

7) That good statistical lrng is an adequate soln. to General A.I. problem: It enables all OZ & ∴ INV probs. Perhaps Fogg. methods are adequate for TM to begin lrng arby info.

100 possible moves for Checkers. fairly good game playable w/o "lookahead" — i.e. a best move for each position. (Theoretically, "Lookahead" is not necy.)

Marvin feels that the statistical pgms haven't been able to go to higher order abstractions ... Also, that there has been much news about what they can do: Much less about what they cannot do.

SN Hvr. A vital point: that the update algm need not be Turing Complete ... it can be relatively rudimentary ... like PPM (or even Lemp. Ziv!). That it is important that the Ref Lang be universal. Perhaps this is equivt. to $TM_1 = TM_2$ — i.e. we get t. update problem to be worked on as a "regular" TM problem ... having as co-corpus, the rest of t. TSQ.

SN In LISP, one can do partial evalns as soon as one has partially completed "expression".
e.g. Sum Mul 6 3 doesn't have t. second arg. for "Sum", but we can evaluate Mul(6,3) & perhaps put it on t. stack. We then have t. nice feature of Lisp using "input" so we can use it as a "3 10", to assign pcs.!

Recursive function Lisp have shortcoms, but take long time to evaluate if stack is used. — We will always try to compile such expressions, so they take much less time (say express them as "for, next" loops. As for t. mechanism of inserting the ("constant" "Boundary Cond") in t. definition [In Factorial(x) = Factorial(x-1)·X; Factorial(1)=1 — t. "Factorial(1)=1" is t. "constant part"! The "Boundary cond" is a better way of looking at it.]

Perhaps we can arrange so that after giving t. main part of recursive defn, t. boundary cond. acquires very high pc. e.g. after writing Fact(x) = Fact(x-1)·X, we automatically write Fact(& we have to complete t. expression w. t. pc of "Fact" set to zero.

8/9/04

4TM

[TSQ] (Ref. umc. design / Ref lang) : 24 } VERY GOOD: Very General! This should be my main Approach to TSQ Design.

30: [SN] Re: Barron's ~1991 paper on ANN and the curse of dimension: I was thinking of counting args to show that the no. of functions possl. would be exponential in the dimension — But this may not be true if one has limit on spectral shape (Barron) or limit on size of derivative or Lipschitz condition (or just $|f(x+\Delta)-f(x)| < A|\Delta|$.)
If second derivative is bounded (perhaps $|f(x+\Delta)-f(x)|^2 < k|\Delta|^2$ or some such other exponent) we have much more stringent confinement of possl. functions.

08 [SN] Re: the VP proof for optimum M+Carlo srch: Go thru it again! Exactly what
7 is p . . . the proby of ?? (10)

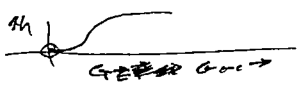
[SN] Re: Update: Update has 2 aspects: (1) updating params of Lsrch guiding PD.
(2) Obtaining new solns to new & old problems using a more powerful (but perhaps slower) Model of induction
0 (.07?) For normal M+Carlo Lsrch w. n cands, time to soln is ~ n. (or 2n or n/2 ?)
.11 For VP " " " , E time to soln. depends on pc of winner.
.12 for Lsrch " " " " " " .
My impression is that for VP, I didn't take normzn into account, & this can be easily done by Lag. Mults or just write in normzn constraint in optzn criterion — I think the Methods the same in either case.
Using Lsrch proby one would want to derive several solns. They would all be biased at very high pc, as opposed to solns obtaind by .11 or .12 (VP).
17 But do work out exact equs. → (199.10) 4

0 [SN] Political Note: Using the/set of current improvements to GP plus Lsrch should give a very fast, powerful problem solving device. How far it could go in a TSQ (or how to write Adequate TSQ's)
22 is unclear. [Set of improvements is " is PPM, (at least terms to PPM, second order correlns) → Use of Search for Common Sub trees: 216.20
23 used for GP; ← Think on this last." Use of (1967 paper) PSG discovery for GP, use of Lsrch for GP . . .

24 [SN] On TSQ's, Ref Language, Ref UMC: So I write the TSQ: Then I write out the hooks (in English) that would be (needed/adequate) for that TSQ: Then I design a Ref (lang/UMC) in which those hooks are (easy to express/"easy" to discover). W.r. tricks of .22, .23, we have a larger class of things
27 that can be "recognizable regularities" — so less limitation on ref (lang/umc). → (196.10)

[SN] One way to transfer info from one Domain to another: Transfer Definitions.
From Macro Context of a problem (Chemistry, sociology, ect), one finds what definitions are appropriate.

0 [SN] I do hope that w. a not very clever update (say PPM) we will be able to eventually do useful TM₁ = TM₂ stuff (≈ [redacted] Phase 2). A possy is that in phase 1 we can obtain good

(say) h(T) functions  And give expected time of soln (for INV probs) or (maxim pE) expected G value after time T. (I've forgotten just how I solved OZ problems "exactly") . . . . $h(T)$ is the pc density that pos. F_2 will get G in time (cc) T for problem $\tilde{G}$ [problem = $\tilde{G}$, T]

[See pp 18-19 of IDSIA Report for details & discussn. In that place ~ pp 16, 17, the discussion of INV probs has special Mode for failures: This is not nec'y for OZ probs. All trials generate data with some G (except if T is too small to do anything!).

Anyway: Ideally, I'd like TM to work these problems & how I work them — whatever I can get TM to work those OZ probs properly will be "Phase 1" remains to be seen.

4 TM TSQ's : Searching for Reps : .14

0 : Hvr, in this last case, t. A.H. hypoth has to be very A.H. : very "visibly" so! — As a result, it is much less dependent on apriPD. So this is a good feature of corr. rel.

[SN] In SAARB ppm for ANL! Not solved! I could do facilities for Definition! CRITICISM: 196.30

103 Def → α BR β ← α is Bare leaves that define a defn. BR is a Branch — a NonTerminal NT. That represents a complete branch. So defns can be for constants or Fcts. After a defn. has been made, it is given a name automatically; its PC is obtained by t. Dirichlet rule (general Lap. rule, using = Carnap's rule ("continuum of inductive Methods"). A (perhaps) way to do recursion. A defn can only refer to past defns. So we first define Fact(1)=1 then Fact(n)= n Fact(n-1) ← No!

See how Juergen did defns, & recursions.

I could do recursive defns then "unroll" them in a compiler to ↓ cc.

→ Look up Peter's work on Prim rec. functs. I think she had a simple, useful way to characterize them. — (Not so easy to find, hvr. — (not in Kleene.)) → 196.25

4 Would it be (possl) useful to have BR in .03 be able to refer to a branch that has already been generated?

4 Seems like Impt ideas! → In devising a good "closeness Criterion" better. Conds:

(1) In writing TSQ's, I will have various heuristics in mind by which previous experience is brot to bear on present problem. Use this for "closeness Criterion"

16 (2) How can I (file/store/access/recall) relevant "close" instances in past? — Would work on "Case Based reasoning" Be useful here? They may have devised some useful filing systems. Perhaps Ask Alice for refs — preferably on web.

Re: (2) (.16) An idea: First use PPM type Lex order to get cases w. same "near context"

0 → α then take that set & somehow look at next closest context, loop to α (.20)

We may or may not have to re "order" (re sort) after each occurrence — but the sizes of t. population to be resorted ↓ rapidly.

~~(3) I'd like to make it~~ possl. for TM to discover (better indexing methods, better "closeness criteria) — These may have to be done simultly.

0: • A disturbing possy: That for rather simple tasks (in Algebra or even in non-math), logical reasoning is needed. — That w.o. it, ssz has to be very large to get any induction at all. — Otherwise, all (guesses/choices) are of ≈ pc. Well, so then In certain areas of induction, we will do poorly w.o. deductive Reasoning. In General, there will probly always be areas in which a given TM will do poorly. — whatever apriPD or search P.D we have, there are probably regions in function space that are & will remain, very distant — Unless suitable TSQ's are provided to suitable bias apriPD & Lsrch.

4TM

00 : [SN] Admin. Note: If I'm in good creative mood. (usually detected by good handwriting (like now)) Then I work on Serious TM problems. If not, do Reviews. The logic is: Reviews need to be done, but should require less cleverness. Also, if I'm in a creative mood, I couldn't get very far into a review before being sidetracked!

→ Also Forth (Juergen's version) makes it easy to implement PD's via a 3 inst func. ≈ "3IU"

[SN] An impt decision is what kind of lang. to use: Present choices: Lisp v.s. Forth. Lisp seems to have nice tree structure for functions, but Forth is much faster, & enables one to do partial pgms befor switching to another branch of the Lsrch tree (This last gives a factor of "tree depth" in speed over "Lisp" ... I assume a Lisp pgm has to be completely specified befor one can begin to run it ... (But this may not be really true!)

0 Juergen's version of Forth seems v.g. (He is not trying to be "interpretive" at all! ~1000 Machine insts. per Forth inst!)

The reason I'm interested in speed, is that it enables me to write TSQs w. larger CJS's ... which is much easier than writing them w. small CJS's. This may be a Dubious "Economy". The main problems seem to be ① writing TSQ's ② designing langs & assoc update Algms for Those T.S.Q's

Another (perhaps) big problem is update Algm for Phase 1. Difrnt update algms give Various (goodness of update ≈ PC) at various cc's for updates. eq (24.33 tells how frequently to use a particular update Algm. That eq. assumes that the update Algms can be "Mixed" (can use one then use another later. Update Hvr, update Algms can do various things. ① They can find new solns to problem old TSQ. ② They can recode old solns to problems (so solns are the same — <This changes pc of cones in the Search PD for Lsrch, but leaves the evaln PD invariant>

Perhaps a very impt Point: When I get to TSQ's & the "final problem" get into trouble, I want to be aware of the Various options possl. in all difrnt aspects of the TM project, that could lead to a soln of the diffy.

One Possy would be to start on a TSQ Now & see what the most dif'lt parts are!

10 : 165.12 : Why would we ever want to look at the Data & use it to control srch, hypoth selection?

One BIG REASON: By looking at data, we enable our Biologically defined system to utilize apri info that we are not Consciously Aware of. This Can Be a BIG Deal! i.e. a very important heuristic. Is there a way that we can enable this heuristic & still have the advantages of the U.D.?

Not so easy! Since we are modifying the reference apriori, after we see the data .... !

Hvr, this "Modified Apriori" need not be v.g. — it can be very Bad!

Cross Validation is designed to deal w. This Looking at the data Apri info. It may not be the Best way, hvr.

Also it is often diff'lt, if not impossl. to apply to many kinds of induction.

Also the: Cross Val; Consider the "leave one data pt out" method. Even it can be made to have spurious results if we make a suff't A.H. hypoth! A suff't A.H. Hypoth can give us any result we like!

8/6/04
4+M

00 : Gen. Discussn, Approach :

Try to get PPM running for Phase 1. T. main idea here is to see if I can use it to deal w. TSQ's of sufft. complexity to get to Phase 2. [The work by t. "Community" on PSGD may be very useful in Phase 2 — I can certainly Augment Stolcke's work by various heurs for word discovery & ngust. discovery. How became to have a nice way to get recursion in CFG's.]

In Phase I I'll be interested in How (fast / effective (= by prob of corpus)) PPM is — Can I get it to learn a reasonable simple ~~How~~ TSQ?

0 A BIG IMPT Q is: Using Phase 1 tech acquest, How do I ~~[illegible]~~ implement t. Universality of t. system (Say I'm using a Universal reference (lang. machine))? *(See .23 - 30)

There must be some kind of "Trade off" betw. using Simple Guiding PD (like PPM) & ∴ having diffy finding using universal msg & ~~hard~~ solve hard problems: ~~vs~~ v.s. using a smarter Guiding P.D. so solving ~~probs~~ hard probs w. t. universal lang is easier. ((Also how t. lang consideration interact w. t. cleverness & t. degree of ~~Skinnerianism~~ Skinnerianism of t. TSQ.

Re: .10 - 11; For every problem, ∃ a soln. expressible in t. univl. lang I'm using. T. Q is: how does PPM (say) help devise such solns? How does it bring regy in TSQ to bear on new problems? If I think a past seq. of prob. solns. is relevant to (should ↓ cjs of) a new problem, this should be implementable via PPM, or somehow t. new & old problem must be recodified so ~~[illegible]~~ PPM helps "connect" them.

23 For a Hard problem: I write a soln in t. Ref. lang. If it takes TM too long to find soln: look at soln. Can I devise simpler problems that have as solns, parts of the soln. for t. harder problem. {By "parts" I mean/ common substrings in solns}. This would make PPM an adequate "Updater".

[In General: when TM is Lsrching for regys in a corpus (or partly coded Corpus), t. regys sought most have a/ low bcost (Acceptably — who coded viz ~~[illegible]~~ what cones found thus far). T. TSQ is designed to make this possl.
All Concs are composed of Sub Concs, so it should be always possl. to ↑ pc of sub-concs before needing to combine them to solve a more diff't problem. — A Big Q: Can I ↑ pc and t? — also what about conditional pc's of those Concs? ↑ This May be a Critical Q.!

30]

(← Mass Back reference (p 190?)):

31: On meaning of "Merging": ~~[illegible]~~ 2 pos (= names = substrings) are pd's over t. set of all strings. To merge 2 of them means to make a wtd sum of t. pd's on all strings: so: 1 (or 2) params to combine them. We then ask: ⓐ How is new ngust it made part of t./ Target Grammar? ⟵ ⓑ Did we thereby ↑ a psip of corpus?

I think ~~[illegible]~~ an answer to ⓐ is: T. new ngust is a NT and we have to find a way to ~~add~~ ~~[illegible]~~ redo t. Grammar around it — How, given the previous applicns in t. Grammar of t. 2 component NT's, I think there may be a standard way to do this.

I guess this Merging will never ↓ t. no. of strs in a lang. It always "generalizes" it. Also, it always ↓(?) pc of corps wrt Grammar (?), but pc of Grammar itself most always ↑, since it's simpler (fewer Components). [I may be using "Merging" to mean stolcke's Merge + Chunk?]

4TM

0 : **On Merging & Chunking as Useful steps for Hillclimbing in PSG.**

"Chunking" simply looks at the Grammar: tries to find common ngrams & compresses Grammar by defining these ngrams. It leaves pc's of derivations intact.

"Merging" — I don't understand how it works exactly! Somehow, 2 NT's are selected & Merged into one NT. The resultant Grammar is supposed to be always able to generate the same corpus (w pc>0), & some merges will ↑ allover (including Grammar cost) psi of corpus

07 on p83 of Stolcke thesis last line before § 4.3.2 (Stolcke) supposed to show that the method has to work!

09 Given target S → AB
→ ASB
A → a
B → b

12 "corpus"

S → AB	(10)	# of case counts
→ AABB	(5)	
→ AAABBB	(2)	
→ AAAABBBB	(1)	

A → a
B → b

So is he saying that (Given enough samples) it must be possible to "unfold" the corpus SS into the target, by merging & chunking (chunking & merging). So there must be a path of (merge, chunk) operations that transforms corpus into target ??

I don't yet see how this must be so.

As new, better rules are derived, each example in the corpus is progressively simplified until we get the final, minimal Grammar. See (31) for promising Attempt!

It looks like Chunking is all that one needs!

Consider the derivation of a case:

$S \to ASB \to AASBB \to AAABBB$

so in corpus we do S → AB, S → AXB, ..., X → AB) → S → AB, S → ASB define Y → ASB, & so S → AB, S → ASB, S → A

S → A(ASB)B ← we can rewrite this as S → ASB

S → AAASBBB → S → AASBB → ASB

It may be that the way Stolcke looks at CFG's is strongly conditioned by his earlier work on things like HMM. The concept he uses to describe his approach to PSGD are perhaps more General than is necessary.

(8.7.04) → When one has a CFG, one can look at the productions (like S → ASB, etc) as an unordered set of strings, & compute their PPM redundance. We can also use PPM to detect implicit links of regularity (i.e. common substrings) in the Grammar ... even tho' the corpus is small. (Tho' for a large "initial corpus",

0 the Grammar can be very large, & PPM could have a reasonable SSZ.

31 Getting Back to (07): Say we had the corpus as (12) & the (true) Grammar as (09), & a good parsing of each s in the corpus with that grammar. We should then be able to convert those parses into corresponding productions (via "Chunking & Merging"). Perhaps it could be done "Backward" by a first getting all productions that were used in the last stage of completion of corpus generation. After this is done, we somehow look for it, until we have it done! It may not have to be done in the exactly right order.

Proof that $T \leftarrow 3T$ is Best for Lsrch -2.0
(rather than $T \leftarrow 2T$, say)

>0 work GP/GA problems.

What is "State of Art" is now, is unclear: 188-32 shou (2003 Dec) may be up to date.
Stolcke (188.30) has web site w. applics of his ideas (on PSG discy, among others)
to various problems & I would be well to see what progress has been made since 1994 (10 yrs!).

Stolcke: Now at SRI & Int'l Comp Sci Inst. Berkeley Calif.
2002 SRILM: an extensible language modeling toolkit.
Stol... 95 .... Computing probabilities
98 Entropy Based Pruning of "Backoff Language Models".

"BOGLE" was sw. in Lisp/CLOS to experiment w. Probistic Grammars & Parsing Algms.

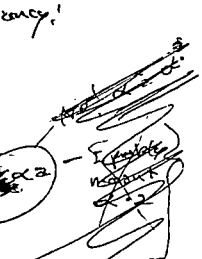
0 SIV It may be useful to try to model a "G" function of a set of strings by a pc-function of the set of strings. I wrote a lot about this, but I think I ran into a dead end because of an error in my thinking about language Models.
This recent exchange on PSG discy may be what is needed!
Along w. the G function I will want to optimize & (at least (param model)) of a $G \to pc$ function like $pc \propto G^{\lambda}$ or $pc = A\, G^{\lambda}$ (A = Normalizing const).
(Here, we are mainly interested in the Goodness of fit of G to pc, in short, by G regression.

>0 : $\ln\left(\frac{T + aT}{T \leftarrow aT}\right)$ Srch; What is optimum value of a? : There are 2 factors of inefficiency!

$\frac{1}{1-\frac{1}{2}}$ factor $= 1 + \frac{1}{2} + \frac{1}{2^2} \ldots = 2$: Its t. amt. of time wasted in repeating trials.

The first is exactly correct; it's not "approx".

T. other factor is t. amt. of time we stand in trials of the final round". I if we assume that t. probby of success is uniform in log domain. Then t. final round will be between α & α^2 — Success can occur at any point. T. expected value will be $\alpha \cdot \sqrt{\alpha}$.



So what value of a give min for $\frac{1}{1-\frac{1}{a}} \cdot \sqrt{a}$? $= \frac{a^{\frac{1}{2}} \cdot a}{a-1} = \max\left(\frac{1}{a^{\frac{1}{2}}} - \frac{1}{a^{\frac{3}{2}}}\right)$

$-\frac{1}{2} a^{-\frac{1}{2}} + \frac{3}{2} \frac{1}{a^{\frac{1}{2}}} = 0$ $1 = \frac{3}{2}$ so $a = 3$

That stopping time for last round should give uniform pc in ln domain: Common situation; related to idea that "1" occurs most often as leading digit in lists of sizes of objects like lakes or towns, etc.

? That t. dist should be uniform in ln domain is easy if we accept invariance w. multiplicative change of scale (dimensionless "stability)

Actually a Grammar Grammar is a v.g. way to get a pop of Grammars.
It enables us to use statistical info on previously successful Grammars.
This is what I did in Sol 64b ... Also J.J. Horning. — But Horning used a mathematician's definition of a soln. to t. Gramm. discovery problem. He may have used heuristic search also, — but not enuf!

PSG Discovery .20 (very specific).

.20 [SN] Gen. remarks ① Does Chow Liu paper just get a better than "Greedy" soln., or does it get an optimum soln.? ② My impression was they implied methods were relevant to "more complex dependencies"

③ On idea of "Order of Choice": If order of choices is known, one can always use that cmp. data to get pc's for each token: Perhaps by PPM. (In normal PPM, order of choice is known: it's simply sequential choice.) Anyway in generating functions, what are "reasonable dependencies"? How is that Q related to idea of "Sub-functions"?

Each App. of a function must be a complete "Branch". Each branch has (historical) "context". This can be (extended) "(external) context" or "Generalized context" (≡ ALP).

In fact. Generation, we may know order (or partial order) of choices, but

.0 we don't have clear idea as to which tokens in the past, our present choice is most controlled by.

HA! In fact generation, a Branch occupies "one place" (in terms of distance from token to be chosen.) So in {add Branch, x}: The context of x is "add"; (by commutativity we could rewrite it as add, x, branch ... which would make it more obvious: ... consider sub branch, x. ; x is the argument of "sub branch"

.0 : PSGD A new way to discover/construct new sets: Say a, b constitute (new set) "α".

a & b have pc's within α. — So we have rule(s) for producing α, then we use pa & pb to code a & b within α. Doing this gives a way to code corpus that is cheaper than using a & b directly, w.o.a.

.25 We can tentatively add (or subtract) a new trend & see if the all over

.25 corpus pc ↑ or ↓ ; I fit ↑ we add that new to α. Similarly w. subtracting a new from α.

We can tentatively construct a new set by looking at all words that (follow / precede) a certain high freq. word. This new set is then "trimmed" viz .25–.26. (modification types / trial types)

Also Note: The "Merge" operation ↑ pc of Generation; T. "Chunk" ↑ pc of model: We can add 1 or more favor of trial types of these 2 kinds.

.0 — About Discovery of CFG's by Stolcke etc. After Stolcke's thesis (in '94) he applied these ideas to much stuff in "A.I." type problems (See his website for enormous no. of downloadable papers).

.32 (Amply data Summary) Shan, McKay, Baxter, Abbass, Essam, Nguyen (6 Fellows) AUSTRALIA (Dec 2003) is review of applic. of probabilistic models to Srch. — But they settled on Stolcke's thesis for their method. These Guys seem to be saying that Chen ('95) & Stolcke ('94) used a poorly defined Apsip function! — That theirs is much better!

My present impression: That Stolcke probably had Apsip evaln. funct that may have been "good enough" to begin. PSG discy problem, & that he had 2 mutation types (Merge & chunk) that were a very good beginning for PSGD. ← (This was in 1994)

In (.32) Shan et al 6 yrs later may have corrected Stolcke's evaln. function & perhaps used improved methods ~~[struck-through]~~ What they do that is new, is to use PSGD to

4TM

0) Baluja 95 compares GA v.s. other methods for 27 oz probs. Other methods "Better".
An ordering of improvement in Prob Models for "GA".

1) PBIL ~~BAH~~ Baluja ~~95~~ 96 — extends scope of PBIL

2) MIMIC De Bonet et all 1997 MIT : "second order pc"

3) ~~Bas~~ Baluja Davies 1997: Improved PE of 2) This uses Chow, Liu (1968) to get improved P.D. They say the signif. improvement.

~~4) Baluja, Davies 98 ~~Optimal dependency trees.~~ "A method of speeding up both PBIL & "Hill Climbing". This is "dependency trees" They Don't use probabilistic Modeling"! Is concerned w. "Which Groups of Params are more imp't for GA". (which GA is not concerned w. — i.e. all crossovers have = pc). Fast prob. models: This paper reviews PBIL and MIMIC. ~~Fast not yet~~~~

Chow, C | 1968
Liu, C |
IT 14 462-467

4) Baluja, Davies 97b (or 1998?) Fast Probabilistic Models for Combinatorial Optzn.
This uses COMIT! How this differs from Baluja 1997a (item 3) is not yet clear.
[P2 col 2 of this paper: ~~A~~ last P supposed to tell what COMIT does. P3 section 2 is a lot clearer.
I think COMIT uses 3) or possibly 2) [which are hy cc methods but v.g.] alternately w.
PBIL ~~too~~ (c) a fast search method. || (or alternately w. a Hill climbing (fast) search.
③ is used to reinitialize PBIL (or Hill Climbing) every once in a while.
My Formula for how often to use hy cc update routines, of 4TMS [124.33] is relevant here: Using methods that give that formula would enable one to tell how much time to allow for a slow search like (3) or (2) & how much cc for fast optimizations, like PBIL (c) or Hill climbing. The, at present, I don't quite understand why COMIT is a good idea (if, indeed, it is!).

5) Pelican Goldberg Survey 2000 | 1999 was latest ref. (~ 11 pp : This is a "lite" review. of Work in this area. So refs to General work on "Building Blocks". ← P5 7/26/04

6) Baluja Apriori Info for Prob. Models to Optzn: 2002 26 pp This looks like a more detailed review (& more recent than 5) ← /P5/7.29.04 7.29.04
Gives list of survey papers: (8 31 39 46 41)
1996 2001 2002 2001 Pel, Goldberg ....

7) Pelican Goldberg Tsutsui 2002: Combining Bayesian Optzn & Adaptive Evoln.
I think this works problems w. simult. discrete, continuous Optzn.

4/31/04
4TM

(This array has case counts for corpus)

.00 : How do we generate cands? : T. ~~[struck]~~ $n^2 \times r^2$ array tells how often each pair of digit values occurred in the corpus. How can we use this to generate cands??

.02 One (slow) way: ~~[struck]~~ look at (~~n plus~~) array: Pick a single element at random, or its total pc. (Cols are imposs.) On basis of that 1 choice, put ϕ's into those elements that are now (illegal / imposs.). The single chosen element, "a" gives pc's for all other elements: chose one on this basis. — "b"

Now, w. 2 chosen elements, ~~[struck]~~ use conditional dist. on elements to a & b: sum the pc's (put more ϕ's in illegal bins). When "c" is chosen, we now have 3 condl prob's on legal choices. We add them

.07 (& normalize) & chose "d", the next choice. We continue until an element for each of the n positions is chosen.

.00 ff: 1) is much slower than original 2) It is not clear that we get the rite pd on cands this way — or even that it is better than the original (short) way.

.0 A Big Advantage of MIMIC over PPM, is that MIMIC can use correlations for non-adjacent elements.

Its Disadvantages over PPM: PPM can use ~~long~~ long sequences (or) for induction, w. hy SSZ, & good induction.

PPM can deal w. variable length strings. Mimic may be able to be modified to do this; I don't know how, yet. [See .18 - .26]

---

15 Some "improvements" of Mimic: (1) use last 2 choices to get pc. for third choice. Needs more memory, but otherwise, it can be designed w. close "analogy" to Mimic. Initial use of ~~xx~~ a permutation of the positions. Use of Max likelihood for choice of permutation. → (.27)

18 (2) Instead of fixed vector of ~~long~~ length n with fixed coördinates; [redacted] make coded system "moving along" as in t.s. predn. The "case no. matrix" has "∞" × r × r elements.

.0 — The ∞ part may be as large as feasible. At (i, j, k) element gives case count for ~~digits i to~~ digit k following digit j by i spaces. How do we use this matrix for (1) predn to generate cands? ~~[struck]~~ (this can (perhaps) be done using the ideas of (.02 – .07) which is of uncertain accuracy. Also, since we have ∞ pts. ~~xxx~~ in the cand, we can generate as long as we like w.o. filling it up! (unless we have a "stop" symbol). So we can continue generating pts until all of the section before the "stop" symbol is filled in.

26 So we have a string of randomized length.

---

27 (3) We can extend .15 to the last [redacted] K choices: we can vary k, depending on current SSZ. (SSZ can ↓ as we close in on "minimum")

.0 → While PPM & MIMIC & its Modifns may do not so bad w. function Generation, What I really want is a method that readily recognizes sub functions

A ~~[struck]~~ Subfunction is obtainable by deleting any complete branches from a cand.

— But I'd also like to ~~[struck]~~ make a funct by taking any branch & deleting ~~[struck]~~ sub-branches from it

---

See if there have been Recent developments in ≈ MIMIC 97 · 184.11 is 1998 ref. "Improved" Mimic" (By Baluja

4TM

(1997 R. Sałustowicz & J. Schm. — Probabilistic Incr. Prog. Evol'n — PIPE)

.10 : [SN] Re: Juergen's paper: I guess he looks t. pc's at t. nodes: Then does a run of trials w. fixed pc's. T. Best could be selected from t. runs & used to modify t. PD's at t. nodes — (Hvr. check this for correctness). He also uses several update methods. } See longer analysis in 5TM 9.11 ff

.04 [SN] Definition of functions: Several Approaches :: Koza: Separate random generation of functions !!

.05 (Juergen) (Approximately / or exactly?) t. same :: Sol: No definition made until ∃ evidence that it is useful: .054

Takes more cc, but They have more pc.(?).

.04 may be one of most critical Q's in AI!

Also consider how Baluja & others deal w. this!

Also t. idea of finding common substrings in DNA, Proteins (M.Li; Gerry Wolff).

.0 "At a higher level": I was assuming that more complex regys would be "exposed" "noticed" at a "higher level" — To what extent is this true? Exactly how does it work? How expensive (in cc a/o pc) is it? What is best way to balance cc (a pc) of discoveries into t. "lower" vs. "higher" levels?

[SN] (Re) Note that PPM is at least as good as Koza branch swapping, because PPM does always recognize complete & partial branches.

Hvr: "Automatically defined functions" (ADF) (.04) By Koza seems like a way to recognize functions that I implement in .054 — at more or less cc/pc. T. Q is gain or loss of cc/pc.

.0 : Comments on MIMIC 1997 De Bonet, Isbell, Viola. (J.S, C.L, P.)

I assume the model is a n component vector w. radix r.

In this case, There are r^2 components of $P(x_i|x_j)$. $(i=1 \ldots r;\ j=1 \ldots r)$.

Here We want to choose an ordering of t. x_i such that $\hat{p}(x_{i_n}) \cdot \prod_{j=1}^{n-1} \hat{p}(x_{i_j}|x_{i_{j+1}}) = \max.$

The $\hat{P}(x_{i_n})$ & $\hat{P}(x_{i_j}|x_{i_{j+1}})$ are empirical values.

So we would need $r^2 \cdot n$ params to describe t. distribution. Hvr, we end up w. a model having only $r \cdot n$ pc. params (No!) (still, not so small). Most pc's would be $\approx$ ~ 0 (using straight rule) & $\frac{1}{200}$ using Laplace rule (200 is size of an "iteration").

.0 We don't want pc's to = 0 since we then can't reach certain values.

Woops! I've been missing a main pt. of t. model! T. model has $n r^2$ params. A complete model would have ~ $(nr)^2$ params. There are nr "states" given by t. r x values at each of n positions, so ~ nr^2 "transition probabilities" (From fact t. "nr" states values need not be sequential).

By choosing a specific "order" (≡ permutation), we only need nr^2 instead of n^2r^2 params. While this & RAM needs, it doesn't help w. ssz: There is no "pooling", so it might be better to store all n^2r^2 params. Hvr., updating takes n times as long also — (But we don't update after each cand.)

4TM Bibl. see 177 for previous Bibl.

Bibl. of papers on "Pop-based-Inc Lrng." "PBIL" & other related papers.
["My name" PBIL v.s. Crossover: Baluja 95]

1) GA & Explicit Statistics: Baluja 1996. | B & J 95
a) Extends scope of PBIL problems where Crossover Useful in GA | Shows HC & PBIL as good as
b) Many Comparisons of PBIL to GA. | GA on OZ problems Seeming to need GA.

2) "Optzn Heurs" | Baluja-1995 22 pp
Exams | 27 optzn problems. (in some detail) | finds GA is not better than Simpler Methods. Examines 7 types of problems

~~3) "PBIL v.s. Crossover – Baluja – 95.PS~~.

11 → 3) Optimal dependency trees – Comb-optzn: Baluja – Davies 97.PS [RS/7/29/04]
Uses pairwise dependencies of params.
Extends beyond "MIMIC" (De Bonet)

4) paper ~ to above (both ~ Jan Boy) pretty similar.

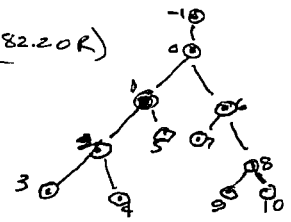
5) Fast Prob Mod(listed els) – Combn Optzn – Baluja-Davies, 1998 | 7/29/04
This method is useful on Both Hill climbing & PBIL.

6 Approach to Prob. Models for optzn: Baluja 02.pdf → good Bibliogy 2002
2002 is latest Ref :(only 1 2002 ref, Grr.)
Looks like v.g. Bibliogy.

7. Baysian OZ plus Adaptive Evol – Pelikan Goldberg – Tsutsui 2002.pdf Pelikan 2002
My name → " Pelikan OZ combining Discrete Contin Optzn.pdf. — 2002

4TM

"Reprise" of (182.20R)



Look at node 5: while in regular Polish, 5 has context 4 (which is not very useful), T. context 1 2 3 4 looks like a useful subfunction

From the approach (181.00–13) the immediate context of 5 is 4; next 0 1 2 further out −1 6 7 2 (3 or 4)

I could code 5 w. a simple "linear" (= sequence) prefix: 1 is immediate, the penultimate node is (0 or 2) (make convention so choice is unique) then (2 or 0) next $\binom{3}{4}$ or −1 or −1 2 $\binom{4}{3}$ etc., so we will have a unique context for each node, of length that can include the entire stack.

If we get a mismatch of any pf, the context stops.

One objection is (for example) that we have to choose where to back: 0 à 2 as context.

Say 0 matched but 2 did not: I'd like to use 0 as context, but if the convention said "2" was to be matched first, I'd use 4. Also, if 2 didn't match, I'd like to

use 1, 0, −1, 6, 7 ··· as a context for 5.

It might be possible to "index" several (!) contexts for 5:

1, 2, $\binom{3}{4}$ and 1, 4, $\binom{1}{6}$

Other than need for extra RAM (maybe available) (18) May not affect run of PC to f.

Prodn. of 2 (!) contexts; ~~If they both contexts were of length k, was past~~

If we have 2 (parallel) contexts, both of length k, then each context will have a certain "Broadness" of its P.D. — The 2 contexts will will have a "Broadness" of $\frac{1}{2}$ or $\frac{1}{\sqrt{2}}$ as wide (∴ better): But a context of length 2k would probably have a P.D. broadness of of much less than that of the k length context. So much more pc for producing contexts of length 2k (as opposed to 2 (!) contexts of length k)

(SM) Areas of induction relevant to our problem: (1) Recent Statistical Mac XIth.

(2) DNA analysis; Protein Prediction ··· (Is Wolff readable?) (May 2c).

Other ideas: (3) PPM with straight RPN or (pure Polish), (4) Modify of RPN like 183 off or 181.0 off (very only) (parent)^k strings. (5) Ivanyuk, Salustowicz (1997) method (what is this?)

2/27/04
4TH

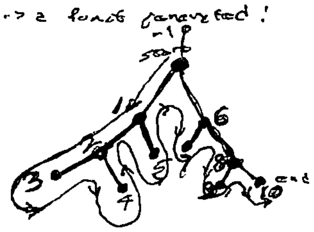
So every finite string starts w. (0 or more ~~Terminals~~ $b = \underbrace{\quad}_{B's}$) : is followed by 0 or 1 fstring : ← A formula(s) = B.
concat.
As a string needs c ~~forms~~ formulas to ~~complete it as~~ xform it into a B.
So, for every string we have 2 integers b and c. (If b=c=0, t. string is a B ← (only b=c=0 gives a B)) ← No — or a null string.
If b=0 ~~[struck]~~ t. string is a fs (function string) : and then if fs > 0, it's a ~~[struck]~~ incomplete function (≡ fi). ← Func, incomplete.
If b=c=0 it's a null string. || If c=0, T. string is a finite seq. of B's.

So, a string is either a seq. of B's (set of arguments) or a set of B's followed by a ~~[struck]~~
fi = (incomplete function). T. inc. func. will have a "c" characteristic ≥ 1. (≡ no. of B's needed to ~~[struck]~~ make it a B.) ← rank

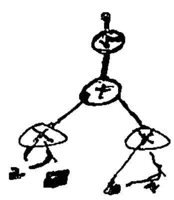
~~A~~ ~~E~~ fi is actually a function w. c arguments.
A string that is a seq. of B's is a seq. of arguments. | So any string is a seq. of args (b: 0 or more) followed ~~[struck]~~ by a a funct of c args. (c ≥ 1). ← possibly

So: ~~[struck]~~ a ~~[struck]~~ notation's utility depends on what one wants to grow.
S'pose one starts at the root: So all incompleted functions will be "fi" type strings: at each point, having a certain "c" value needed for completion. Hvr, since we will be usually using "partial" matching, t. string to be matched could start w. a few "B"s.

How is a funct generated? This is t. "idea to t. ~~[struck]~~ hand wall" again.



Consider a funct like $\sqrt{a^2 + b^2}$



√ + × a a × b b
Sqr add mul mul

Also would t. utility of x^2 ... a unary function be easily found?
× a a : perhaps leaves (~~[struck]~~) could be addresses ~~[struck]~~ for argts.
One could use a ~~[struck]~~ "definition" facility, (but this misses the point of using PPM.)
Try finding what kinds of useful functions are found by PPM.
Recognizing that × a a (≡ x^2) is a unary function maybe the "higher order" intelligence.
To some extent, x^2 is "recognized". When × a occurs, a is more likely to follow.
— But if × (Bstring) occurs, would same Bstring be likely to follow?
I guess I want to find out just "what kinds" of regys PPM could find in "RPN" or "PN".
Could PPM find useful regys in "infix" notation?

⟶ Using just the (parent(s)) of ~~string~~ relationship ~~of [struck]~~ of function trees; PPM could find useful stuff.

There are probably many ways to code function trees: Certainly some (w. PPM) will be much better than others.

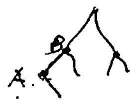
Actually, 181.00 — 13 may not be so bad! see 183.00 Spec q.v. ⟶ 181.00ff

7/27/04

4TM

> Advantages of Lsrch over normal GA:
> 1) More efficient: less LC wasted on repeating hyp. trials
> 2) Ability to deal w. hyp. trials in a goodway — so it can deal w. recursive funct. easily

: In Applicn of PPM to Generation of function trees:



To start off - each possible funct or Terminal (≡ leaf) at a node has an uncond pc. First uniform, then Laplace Rule.

Next, it depends on function at a node.

So sprouts at A have a pc d.f. that depends on A.

Next, it also depends on parent of A (≡ B) or on other sprouts of A.

So in general, the dependency works backwards & sideways. For each node we can list nodes that are "distances back" of 1, 2, 3, 4, etc.

There is some Q about whether "matching" deeply in 1 direction is more valuable than a shallow match in another direction. We do have a proto-ordering of matches.

At a given node, can we have > 1 possible "history"? If so use suitable wtd mean pc. for children. → 182.37 183.00

It "context" is defined only by parents, & parents of parents.... then each node has a unique seq. of parents leading to the root — so we can put each node in a "lexical order" & use PPM & its variants.

Look at the (Schmidhuber) paper: Compare his method w. ooff & .14ff.

Also, compare use of PPM on the RPN string expression of functions that I was originally considering. Do the new ideas (of ooff) really have any advantages?

In RPN any complete Branch (including all its leaves) is expressed as a coherent string (ngm) — so PPM may be at least as good as Koza's "random branch exchanging"

In RPN, a string can represent a complete branch (see .20-.22). Also, if it starts out w. a function. Maybe not complete.

It can represent a branch plus zero or more branches plus a "partial branch."

Recursively define: a string can represent sequence of 0 or more branches terminating in a "string".

(complete defn of "fstring")

→ A fstring starts w. a function, can represent 0 or more branches followed by a "f string"

A "t string" starts w. a terminal (≡ leaf), It can be followed by 0 or more leaves, & may or may not be followed by a "f string".

F T T F F T F

An ngm can start w. a funct. — It can complete the funct & become a "B" (≡ complete branch)

or, it can be a part function so all args aren't complete.

A "Pfunct" is a string starting w. a function, but not having all args of that funct.

Partial defn. → A Pfunct consists of a funct. followed by 1 or more B's (≡ complete branches) followed by a Pfunct.

For any Pfunct, it is possible to convert it to a "B" by concating on one or more B's.

[illegible] Pfunct has an assoc. integer c, (≥ 1) that tells how many concated B's are needed to make it a "B" — ≡ "fs"

Every string beginning w. a funct., has a c no. (c ≥ 0) that tells how many B's are needed in concat. to make it a (complete) "B".

Every string not an fs, must start w. a T (terminal). It will have b terminals before first funct. occurs.

179.29
00 .179.40 : say we have s trays of length ℓ, from an $\ell \times \ell$ grid.

Every time an example occurs, we have ℓ^2 updates of params. Each of the ℓ^2 pts has d^2 possys to add I to.

At the end of all updating, we have $\ell^2 d^2$ bins that have integers in them, that represent the population.

Given $\mathrm{Poly}(\ell d)^2$ array, I don't know how to use it to generate Mt Carlo trials, or to list trials in Y pc order.

07 One possl. guess at McC. trial generation: By suitable summing over $(\ell d)^2$ array, we get pc's of each symbol at each of the ℓ points. How choose a first symbol at the (actually its at $\geq (1,1)$ pt in the array) pt with pc = its marginal pc at that pt. Given that symbol at first ℓ pt.,

10 we get the pc's of all symbols at other $\ell - 1$ pts. directly from the $\ell - 1$ data pts at the assoc Matrix Elements.

11 This is like some kind of Mt Carlo approach....

12 How good it is .... I'm very uncertain!

13 Assuming .07 — .12 is ok., to generate cands in Y pc order!

First pick a "a hypc cand" in poly way: first pick the most likely symbol for each of the ℓ positions. Then the product of pc's of these ℓ symbols is P_0. This may will be used to calculate the initial limits for search.

Another way to get a P_0: pick the most likely symbol at all positions. Given that symbol, pick most likely co- pick symbols of all other $\ell - 1$ pts w. most likely pc's given the initially chosen

20 symbol. These $\ell + (\ell - 1)$ choices give another way to get a "P_0".

Many procedures obtained a P_0; we try all cands in pc order. $pc > P_0/2$.

Then all cands $pc > P_0/4$; $P_0/8$ .... Cf. use of a factor of 2 is not least: factor of 3 may be a bit better.

These pc can be assoc w. trials viz .07 — .11 ... Note .12 !!

Other selection ways to associate pc's w. cands can probably be devised.

→ Since $(\ell d)^2$ can be very large: say $d = 20$, $\ell = 10$; 200^2 array: Means our SSE for each pc. is quite small. Suggests we use a variant of PPM.

General Remarks about GP, PPM: My impression is that PPM (& similar assoc Methods) are not much good unless SSE is very large. It suits my prime GP, but not by an enormous Amt.

8 The "quick/rug" of Humans in certain situations may not involve a fairly large amt. of (internal to human) search. The idea of writing good TSP's is to try to find ways in which the path to a soln. is, indeed, of hypc, & design (or guess) & assoc. search techniques that make these paths feasible.

4TM **SM** : Density Estimation ~ "Nonparametric".

0 : Paper: IE³ Sys, Man Cyb: (Part B cyb.) Aug 04 pp 1708–1717: "Sparse kernel density construction"... Is concerned w. obtaining a density estimate for any $\vec{x}$ point, given data pts $[x_i]_1^n$.
They end up using a kind of Gaussian interpolator; They say its relatively fast & accurate.
[The relevance to SM: Given n data pts of contiguous of k-1 driver values & 1 driven: To estimate density for predn.]

07 [In same issue of Journal: pp 1730–1742: Discussn of ways to improve mut/cross in GP. Looks worth reading for suggestions.

> An Arch for Missing Parallel compact GA: PDF D:\PD\ 7/25/04
Also see desc of "compact PDF : IE³ Ev. Comp Nov 99 pp 287–297 — D.E. Goldberg.

12 Given a population of N binary vectors of length k (N·k bits)
we can represent the population as a vector giving the pc that each bit is a 1. (no interaction between bits) or somehow as a Multivariate D.F.
That does consider crossovers. The cross cov would be a k×k symmetric w. $\frac{n(n+1)}{2}$ params. We can also give 3rd & 4th ... order corr. Matrices.
I had a computationally neat way to express correln Matrices of this sort.
For many high order corr matrices, each time a case occurs, one can update all of the relevant matrix elements.

Note: In the more recent paper (.10): They are using Mt Carlo directly to generate trials...
Looks much better! (It was randomly for these models of the Quality PD.

In .12, instead of binary elements, say we have radix 7. Then correlation between the i-th & the j-th component of the correln matrix will be a cell in a 7×7 matrix (49 elements).
Each time a case occurs we put 1's in various array positions.
We may find easier ways: the matrices are sparse & we only need to know total population of each element — so there may be much more efficient ways of storing the info: Maybe hash tables, or an analog of the way the BWT or PPM does it.

.29 → 180.00

30 (SM) Quick Notes on Pogg: Assuming large Many each needs: If we did a dim correln matrices we would somehow get a Quadr form & a Multivariate d.f. Getting Ens & Elfs, we Set of unit-length or arbitrary length vectors as "multip descriptors", we could get Multivariate d.f.
For MonteCarlo Guesses (as in ordinary GA), or try to get them in pc order as in LSrch.
Getting Multivariate choices in pc order into bodfft. — The arbitrary set of binary vectors makes it sound like getting them sets in pc order (which is somewhat harder).
— I think this is done: List all cands w. pc > ε. Then list all cands w. pc of $\frac{\epsilon}{2}$ then $\frac{\epsilon}{4}$ .. $\frac{\epsilon}{8}$ ....

→ 180.00

ATM vis SM vs ALP.

Cross Validation:

To: : We have a data set $[x_i; y_i]_1^n$ we want $y = f_\alpha(x)$ as a good approx.
α is a param set. α_j is the best fit α for the data set $\{x_i, y_i\}_{i=1\ldots n}$ w/ j^{th} data missing.
So error estimate for large n is $\frac{1}{n}\sum_{j=1}^{n}(f_{\alpha_j}(x_j) - y_j)^2$
i.e. we use param α_j to predict y_j & look at error^2.

The variance of this measure is not easy to obtain, since estimates are highly correlated.

To apply to SM problem: this is not so easy, since eliminating a data pt. may not be clearly defined, i.e. eliminating "1 trade" depends on which TA was used since each α tells when we go in & out.

We might just arbitrarily cut out sections that seem to be of average "trade length" time. We terminate a trading seq. on trade before it would have to continue beyond the boundary. However, we can always start a trade at the edge of a section,

However, consider A.H. hypoth: $[\text{IF } (x_i \text{ occurs, guess } y_i)]_{i=1}^{n}$: ← No adjustable params.
If all x_i are distinct, this will get a perfect score [via cross val. test] it will not extrapolate well at all to cases beyond the data. We could make the rule of (1) more general by choosing a y_i from the x_i that are as close to the input digit x_j as possible — so a very A.H. "Nearest Neighbor". If we have >1 nearest neighbor in the data set y_i's ... use a uniform p.d. over those y_i's for the prediction.

→ So it looks like Cross Val. does not deal w. A.H. hypoths. (Perhaps Cross Val does this too!)

Well, ALP would not do better, if this were the only hypoth being tested! However ALP normally does include the "programming" Pem (as well as the above All Pem). — Both have about same wts.

The A.H. Hypoth that uses "nearest neighbor", is probably better than the A.H. hypoth for the prediction. Also, ALP would give a warning that little of any compression was obtained from the A.H. hypoth, so we should expect poor prediction.

Reasoning: ALP doesn't do anything that tells how much better one hypothesis is than another model. If one is doing Lsrch, then to use "simple length" for a prob is not biased. But for Practical ALP(Lsrch), we use a guiding p.d based on previous probs solns, which is certainly "biased", but perhaps not badly! T. Q is: would it give unbiased p.d.'s?

Another "cross val" method applicable to Sequenced data (like SM strats):
Given data seq: $x_n \equiv (x_1, x_2, x_3 \ldots x_n)$. For $j = 1$ to n do predict of x_{j+1}, using model made on $x_1 \ldots x_j$. The error will be e_j^2. e_j^2 will be a fn of j. Smooth it to extrapolate into future. As a simplistic model use $\sum e_j^2 \equiv$ constant. (☺). This may not be legit ☺. In fact, it is wrong, even if we used full(uncomputable) ALP! Make better model: say $e_n^2 \approx \frac{n}{k+n}$? or $\frac{n^d}{k+n^d}$ $(d<1)$.

See 179.00 for a paper that discusses cross val. and "density estimation".

7/19/04

4TM

I have copys of all of these: usually in full hardcopy, some only partly in hard copy
all are in PS directory (HP computer).

Bibl: see 184 for next Bibl.

: Bibl. of papers on PPM & assoc ideas:

1) Unbounded ~~Context~~ Length Contexts for PPM. Cleary, Teahan, Witten ~1995 (has 1994 ref)
Reasonable explan of PPM. Shows how related to BW-xfm (Burroughs-Wheeler) ← 1994 ← [PS/ — P28 6/6/03: Trace Comp. Leda Cleary]

2.) A Block Sorting Lossless ~~Many~~ Data Compression Algorithm 1994
Burroughs and Wheeler.

3) The Zero Frequency Problem. Witten, Bell IE³ IT July 91.

4) Solving ~~the~~ the Problem of Context Modeling: Charles Bloom ~1999

5) Modeling English text: PhD ~~thesis~~ thesis W. J. Teahan 2002 ; 7/24/02 ← PS/ : HP

6)
Balkenhol, Kurtz, Shtarkov 1999 Improvements on BW xfm. Is concordant.
Speed & RAM needs.

7) PPM one step to practicality ~~[crossed out]~~ Dmitry Shkarin ~2002 : ~~[crossed out]~~ High claims.

8) Bell, Moffat, Witten ~~≈~~ 1995 (in PS/1.7.04) : Hardcopy of P28:
Reviews how good compression was at that time. ~1.3 bits/char for English for known compressors.
We now get ~2 bits/char for large corpus ~ PPM (BW xfm).

9) Bunton 1996 : PhD thesis (PS/CHB 7/19/04 → PDF : Buggy) ← only 18 pp : computer journal 1997 — says get 5 to 12% improvement. PS ~~[crossed out]~~ weighs less buggy — but 2 columns. * Semantically Motivated Improvements for PPM variants. " I Made Hard Copy ← Mike Suggests to front of

7/18/04

4 PM 3:14

[1879 to 1910 or from 1849?
or 1869 " " ?]

0 : Here, We will be interested only in the symbol probabilities and in how they were obtained.

SN In the processing of $\vec{L}$ via the kernel k: The wt. of depends on $k(x)$ $|x|$, $|x|$ being distance from the center of the kernel. Hvr, in fact, the wt. should depend only

24 on the number of matched symbols betw. the "Matchee" and the context considered (e.g. "Match Length").

This would changed discretely, integrally, not continuously like $\sum_i a_i e^{-b_i|x|}$.

Would this make much difference in result?

Instead of a continuous kernel, one mite have a function of integers only, they integer being the "Match length" (.04).

---

0 → 7/23/04 ≈ A way to do this: For each "Length of match" (w. present corpus to be predicted) we have an assoc. wt. This set of wts was obtained by optimizing over "distant past" — it is a kind of "updated" apriori. There may be a relatively simple (approx perhaps) algorithm for updating the "apriori" (≡ the wts). Every so often we will have a length of match > any ever before — so no "apri" data. We could just not use that data until a suitable wt for it had been found ....

5 Alternatively, we mite "smooth" the known wts & extrapolate for the "novel" Length of Match.

---

16 Aux info mite be the amt. of info in the predn — or, more exactly, how fast that info is w. the particular symbols that are 'related' from each context.

So we have the longest context "abcde"; the next longest is "bcde". The ratio of aprps would be the pc of symbol "a". Also relevant is ssz of each context.

0 Say Prefix abcde has occurred k times. Using maybe Lap's rule for pc's of

.21 predicted symbols: compute product of those k predns by context abcde.

.22 Then compute same product of k pc's for same predictions by context "bcde".

The ratio of these 2 products is part of the relative wts of the 2 contexts.

Another factor is the pc of symbol "a" which is the ratio of aprps of the 2 contexts.

wt. ratio of $\frac{abcde}{\text{to } bcde} = \frac{.21 \cdot (\text{pc of symbol "a"})}{.22}$ ← I'm not at all sure of this! How pc of "a" should enter is unclear!

Is $(\text{pc of a})^k$ proper factor?

Hvr, I suspect that .16 ff may not be very impt. It may give a few % in entropy/symbol.

— but what is of more import in GP, is finding reasonable pc's to items, ngrams, etc.

10 — The sort of thing discussed in 179.07 paper.

4TM

0117440 : — (see section 1, P7, eqns 3 and 4 for fuller discussion).

z_0^j is the a priori probability of O^j and is associated with the language used to describe the function, O^j.

In Phase 1, the system looks at the set of O^i functions that have been successful in the past and obtains from them a probability distribution on candidate O^i's for new problems. PPM is used in the present implementation to obtain this probability distribution, which is, in turn, used to guide L search in finding O^i's for which equ 1 is large. This probability distribution on O^i will be quite different from z_0^i, which is a function of O^i only, and <u>quite</u> independent of any information about past successes of various O^i's.

A telling criticism of the Phase 1 methodology, is that it does not really <u>understand</u> that it is trying to maximize equ 1. It just makes trial O^j's "similar" to previously successful O^j's and uses equ. 1 to evaluate them. It does not understand why certain trials are better than others and has no way to use information in unsuccessful trials. In spite of these deficiencies, we expect that Phase 1 will be good enough at induction to bring us to "**Phase 2**", which <u>does</u> have a good understanding of optimization and <u>can</u> use information generated by both good and bad trials.

Before telling how PPM is used to generate trials for Lsearch, let us first look at how it is used in text compression. To compress text, PPM tries to predict each symbol of the text, based on the "context" of that symbol; i.e. the sequence of symbols immediately preceding it. Here, we mean by "prediction", that PPM is able to assign a probability to each possible symbol that might occur at a particular point in the sequence. The product of the assigned probabilities of all of the symbols that <u>did</u> occur, is a measure of the compression of the system. The negative of the log to base 2 of this product will give the number of bits needed to code the sequence.

We <u>can</u> use PPM for hill climbing <u>if</u> we can update using graph (Good) cond. Hvr. older taken pc's can't be changed after solution: in T≤2T, we can only update fully, after each T≤2T "Round"

4TM

dup

: This is a rewrite of 159.20ff!

Title

The Application of PPM to Phase 1 of Alpha.

Introduction

We will show how to use the PPM text compression program in the implementation of Phase 1 of the Alpha, machine learning program. First we will explain how Phase 1 works and how it differs from the more advanced, more general, Phase 2.

The PPM preset predictor/compressor is then described and we show how it is used in the L search part of Phase 1. There are several ways in which PPM can be implemented — we will discuss the trade offs for some of them. L search can be done by either limit doubling, parallel search or Monte Carlo search. In the present trials, we will be using limit doubling & we will explain its advantages for the present application.

7/18/04

4TM

0o: This is a rewrite of 159.20ff!

Title: The Application of PPM to Phase 1 of Alpha.

Introduction: We will show how to use the PPM text compression program in the implementation of Phase 1 of the Alpha, machine learning program. First we will explain how Phase 1 works and how it differs from the more advanced, more general, Phase 2. The PPM predictor/compressor is then described and we show how it is used in the Lsearch part of Phase 1. There are several ways in which PPM can be implemented — we will discuss the trade offs for some of them. Lsearch can be done by either limit doubling, parallel search or Monte Carlo search. We will explain the advantages of using Limit doubling for the present application. } changed.

A critical part of the Alpha System is the training sequences. We will describe various methods of implementing them.

Section: What is Phase 1?

Phase 1 is described in section 7.2 p.24 of Sol 03b (NIPS paper). The AZ language is used to both describe candidate functions, and assign a priori probabilities to them. In the implementation of Phase 1 that we will describe, we may use AZ or some other languages to describe functions. Associated with each language, there will be a natural a priori probability distribution. — This a priori probability will be used to help evaluate the "fitness" of candidate functions.

There is a different probability distribution on candidates that tells how close the candidate is to successful candidates of the past. PPM will be used to implement this distribution.

The problems of Phase 1 are all "Q, A" type problems: (see section 1, p 6ff of Sol 05b for description of Q,A problems). Q, A problems are a fairly general form of induction problem. The main problem in Phase 1 is always to find functions O^j such that

on line by itself. → $a_0^j \prod_{i=1}^{n} O^j(A_i|Q_i)$ is as large as possible. (different line)

Note: There is a discussion of this as eq (3) of Sol 03b. ↑ page?

7/16/04
4TH

: My conjecture about Barron's result on (≈) ANN-like models: That while they used fewer params to describe model w. given/precision, they ~~sumits~~ require ~~[illegible]~~ more precision, so perhaps same total no. of Bits.

Having fewer params m'ke ↓ cc of finding Soln.... but maybe not: If one looks at random, search time depends on no. of bits in soln, indep of no. of dims.
If hill climbing is being used, cc of soln will depend on hypervolume of "Basin" that contains a soln. (T "Basin" is a region in which all slopes toward desired soln.)
Maybe Basins are smaller for hy precision Solns.

For Induction, its just t. no. of bits in t. soln, not t. no. of params, that determines t. wt. of t. particular Model.

Hvr, this Barron result is weird! The logistic function ⎯∕⎺⎺ usd has no params., so t. wts must have all info. That t. net is able to describe ~~functions~~ w. few params is amazing! T. Kolmog thm about expressing a n dim funct w. several 1 dim funct, uses very complicated 1 dim functs (as "Basis"). — Barron does not use that trick! T. Q is: is it possl.? Increases in precision would not seem to be adequate!

At any rate, it may be possl. to show that one needs lots of bits to describe ~~n-dim~~ functs ~~w~~ (precision ∈) using logistic functs (of no params). This could be done by Counting no. of n dim functs w. precision ∈ (satisfying a certain constraint) for that ~~[illegible]~~ ≤ x[illegible].

.: Google! Today I was surprized to find that Google (which says it has ≈ 2^{32} web pp (≈ 4×10^9)) can do PPM.! I.e. if I wrote t. sequence "acabdfg" for Google search, it would ~~[illegible]~~ read all 2^{32} pp to find that sequence & present results to me.
Note google on acabdfg ... it reads all 2^{32} pp. to get this. Takes < 1 second.
~~[illegible]~~ We could then get Google to generate "Google random" sequences.
Maybe ramblig strings. Also Google random functions — to be used to solve problems.
I don't know if Google ~~[illegible]~~ will read pages that are pure LISP, or "C" or Fortran. — Try this out. For certain ~~[illegible]~~ sequences, there will be too many refs. Google will ask for ~~[illegible]~~ further constraints. We can write!
{"acabdfg" LISP} to constrain it (initially) constrain it (?) — Other ways are possl.... say by defining Language, Google has other constraints typed in "Advanced srch", "Macro Contexts".
Also, look at new Srch eng ~~[illegible]~~ BLINKX ~~[illegible]~~ 7/16/04
Maybe Get Lew Mortan to write ppm to do this on Google!

SN On improving PPM w. definitions: for an ordinary linear corpus, we make up definitions, & ever so often, we reparse corpus wrt those defns. [166.11 ~~[illegible]~~ ff describs a possibly easy way to find useful ngms.]
Hvr, when we define function (subtrees), we also must periodically reparse t. corpus wrt. those defns. My impression is that finding these common subtrees is not so easy (See 165.20) — that may be possl. to devise something like PPM srch, because PPM srch can be described in terms of "tries" & subtree srch seems to be a trie srch. [See 167.385 ff for stuff in this spirit]

Also note that subtrees often can be expressed as ngms! (167.18 discussn)

(actually just about any large enough, "well defined problem")

: On Preventing (Ego from Emerging (Awareness of outside world") in TM!

Any open ended problem — like OZ probs ... induction ... or even INV probs, are potentially capable of having "Self-Improvement" (≐ SI) as a subgoal.

This will occur in any "large" INV problem (i.e. Top goal is soln. in min time) or in OZ prob. w. large amt of cB. — In such cases, it will be worthwhile for TM to spend a fraction of its available cc on **SI**.

Working on SI in a TM that can use resources of "outside world" — such as Internet is potentially capable of realizing outside world can be "Manipulated" (as can the USR). [≡ "Marner"].

We may be able to limit TM's internet access so that it can't manipulate outside world.

One way would be to give it a "copy" of the internet that is not connected to the outside world — Say one of Brewster Kahle's archival copies of the internet. Unfortunately, it would not be able to use Google, which & similar services ... which cuts down its value considerably — but eventually, it would be able to simulate Google's functions (if it had enough cc ... Google has maybe 100k computers) ($3000 per computer, $10 M buys $\frac{10 \text{ M}}{300}$ = 30k computers!)

Hvr, Google may be "effectively" simulatable w. much less cc.

We might give TM some kind of "Limited access" to Googlish services ... otherwise, it could easily set up its own web and manipulate it in other ways.

→ [It may be that it is quite diff for TM to do much in "outside world" by access to people.]

: Getting Back to the Sheep; review of 158.00 → .40 → 160.30 — .40

Best and cheap review, w.o. much bibl. refs.: Then put in bibl refs later by scanning back thru notes: The review will give places to bring references to relevant discussions.

7/14/04
4TM

yurat@[illegible] ku.edu.tr

SM .20

0:170.40 : not codes for t. corpus, but over PEMs. : This makes Biased searches more dif'lt to design, but still not imposs'l. Is there a way to define "unbiased srch" that will not discard some [illegible] possibly impt. (good/PEMs)? Note, t. srch is over partial recursive forms.

Perhaps what we want is a srch technique that has a "shut down" (≈ Ly approp).

The "Bias" in srch can correspond (perhaps) to a bias in t. U.D. itself — i.e. like a pri. info.

.05:170.38 [SN] on objections to Occam! PS 6/13/02 "Against Occam"

If these guys found a short code for corpus, then found a longer code for module w. better predns, then it's likely that BC of model + BC of data was shorter (for the last) than t. model w. t. "short code".

— But look carefully at just what the (a) claim (b) actually did!

0:168.90 : [SN] [SM] : Turn on Schmid's "Max flatness for ANN's SM)": For k dif'nt interacting stocks, $[S_i]^k_1$

Find a ANN function ∋ $F(\vec{S}) = 0$: To do prediction, Fix k−1 of t. known S_i's; use them as inputs to F( ), then try various inputs to other S_i: It gives a D.F. (not necly about 0) but this D.F. could give t. (inverse) d.f. of predicted S_i.

This uses fewer params than having k−1 drivers for each S_i! — But t. trouble is, we usually have k−1 S_i values from yesterday & we want t. best S_i from today — So this will not be useful!

→ Hvr, we can have one ANN w. k inputs of yesterday's values, & k outputs of predicted values. We might then Minimize (≈ sq error) averaged over dataset.

→ We should be able to do better than Jerzy, since we don't need Tng. set & test set.

→ Also, we could do ∥ models of ANN, which might get signif'ly better predns.

— i.e. J. used single central "best point" in param space having max (min) Hessian.

I would use all pts in space around t. "best pt." — parallel models.

→ Also, we don't have to keep our drivers in t. set of S_i to be predicted ... Tho if we use same set of drivers for all driven, this does restrict us: — In general, t. no of drivers will be >> t. no. of Driven: So we [illegible] use a single ANN for all predns.

→ Also, we would be changing t. set of drivers, from time to time. (The Soy problem enters here) Also we would change t. set of driven from time to time!

Each day, we update (x windo) the {set of drivers × set of driven} array, giving effectiveness of each driver for each driven. T. driver set is >> t. driven set. Any thing can be a driver, but t. driven have to be "suitable" (large vary, large sums of money, large volume).

The updated array gives good cands for modfn. of yesterday's ANN.

7/13/04

4TM

00: : types of Mathical problems (like in Phase 2 of α)?

10 : SN Opt. value of using (Kullback-Leibler error in Thm 3 of Sol 78; first, this larger than $(\text{err})^2$ (K-L error)) so more signif. but ② If the alphabet is very large so p_i's are often very small, then for $\sum \frac{(p-p')^2}{(\ldots)^2}$ to sum to $< \alpha$, say, is not such a big deal, since p_i's are very small. However, that $\sum_i p_i \ln \frac{p_i'}{p_i}$) converges in this case is of Much more interest!

In fact, in "Bayes induction" problem w. corpus of unordered, finite set of finite strings, we can use the set of strings as an alphabet — in which case the true generator & ALP give

.19 p_i & p_i' resp., in which case $\sum_i p_i \ln \frac{p_i'}{p_i} < \ln \frac{Alp(\text{corpus})}{P_{true}(\text{corpus})}$.

0 — In :19 we have $\sum p_i = 1$, so this eq. has to be modified a bit.

— Also in .19 that eq. corresponds to predicting only one "Token": unclear how Thm 3 of Sol 78 applies.

23:165.18. SN On Occam's Razor: If α & β are 2 PEMs & P_α, P_β are both codes for the corpus, including aprioris of α & β. — If $P_\alpha > P_\beta$ then by corollary of Th 3 of Sol 78, P_α has smaller expected ms error for the known corpus.

[My impression is that Bayes (Because of the peculiar properties of ALP) implies that P_α will probably do better than P_β in the immediate future. See 164.30 – 165.18 for possible ideas (164.30 – 40 mainly)]

on this. Perhaps if we assign aprioris to α & β properly, the discussion will be as if α & β were

5 Chosen Before data was seen. If a_α & a_β are aprioris of α & β resp., Then, w.o. seeing corpus, we would (perhaps) assign apri. wts a_α, a_β to the 2 codes.

At present, This is a unsolved "Unsolved Problem".

See 164.30 : It seems relevant, perhaps identical!

So 3 probs: maybe identical solns!

1) In approaching the U.D., how do we know one model is closer than another to the U.D.?

2) 164.30 : How does ALP get rid of the "trng set, test set" diffy?

37 [For Occ razor: see PS 7/14/04 for ~~review of Occ Raz~~. ← (1998)

38 → 3) Occham's razor 170.23, 165.17 | also 6/13/02 ("Against Occam" ← (1996) → 171.05 for comment on this)

·9 [4)] (Not solved) In induction, increased ratio of p_i's of (several) 2 future events. How can we find

40 these in an "Honest" way by summing ... ? Recall, normally ... → 171.40

7/12/04

4TM

0: 168.40 Methods that are "continuous" — we can spend arbly large amt. of times, but, presumably w. "diminishing returns". ["continuous" in amount of cc used in them & amt. of ΔG expected from them.] The arg. leading to eq. 124.53 can probly be used to tell how much time to spend on updating.

The "Mixed update techniques" may not work together! Suppose uses a slow method to devr various sub-functions & ways to put them together. If this slow update is followed by a "fast" update ... like PPM, we lose all info in 1 slow update — tho t. next slow update can build upon t. previous slow update.

A way to Usefully Mix t. slow & fast update (PERHAPS!) After slow update" derives new "grammar element" & compresses corpus by recoding it using those grammar elements" — PPM can be used to further compress that code (sometimes?) — At any rate, we mite use PPM to "interpolate" betw. the "slow updates".

→ Or, we mite have the alphabet defined by t. previous slow update; code t. corpus in terms of these defined items —

So; each slow update creates a new Alphabet for PPM updating.

0: 168.29 Alife: Problems in TS "identifn". We are given a space of sequence generators; containing random elements & several unknown params (some nonlinear, say). To identify t. seq. generator within a certain precision. [t. params can be chosen by each contestant for t. other contestant ... or chosen at random for both (a common prob. for both).]

.20 is perhaps closer to t. predn. probs I want them to solve.

The "Contest" to get max compression for "Calgary Corpus" is a problem of .20.

So in .20 ff, t. winner could be t. one that best predicts t. sequence. The space of models of the Corpus can be quite large, but it has to be stated. ~~The no. of params~~

The size & nature of t. space may be such that identifiability of t. generator is impractical, & that only approxns. are feasible — & can be used for predn.

— which is much like t. General Predn. problem [We can generate Stochastic T.S. using random discrete params & random continuous params — T. "God" could give them as part of TS]

It mite be (possl. feasible) to ... 2 or more McTiums to pool their talents in trying to predict a sequence. Could there be a way for them to share their "Secret tricks" to create a prediction scheme better than a simple linear combination?

{There has been much work on optimum ways to combine preds of the diff. expert predictors ... I'm not interested in that ... I want the Cands to share techniques, tricks, sub techniques.

Well, what about a lang. for dscribing soln. techniques for time series or for other

7/11/04

4T

ALIFE → "Micro Cosmic God" .00

.00 (Spec 149.19) [SN] on Alife w. "Micro Cosmic God" How do we structure t. Alife Society so it can usefully work toward solving difft problems? I thot of t. Middle Ages; in which Mathematicians would pose problems to one another: Trouble is its easy to make difft problem that t. proposer can solve easily but the "acceptor" of t. problem finds difft. e.g. factor product of large primes; solving alg. eqns.; perhaps combinatorial problems,

Hvr., If a problem is of that type (≡ "trapdoor function") no "acceptors" will be found! — so: Example: proposer says: "I have th. product of 2 primes and t. result is 200 digits long. I will give you t. no. & if you factor it in

10 <1 day you get $100; If not, I get $100.

Consider eqs $e^{ax} + \sin by = 0$; $\ln cx / edy = 1$

If I give you a, b, c, e [limits on range of these params] can you get soln. for x, y

14 in 1 day to .001 accuracy? | or, abcd are selected at random; Both Mathns try to solve it; first one wins. | or, each is allowed to select params of problem for opponent Mathn. → (.25)

15 To push tech skills of this culture in certain directions, we have th. frequent funding TSQ of t. "Mcosmic God" — that gives very large prizes for solns of its problems: Individuals may solve them; or coalitions of individuals may organize to solve those

18 probs, & divide up t. spoils in some predetermined way.

.15-.18 could be t. only source of money in put to t. economy.

20 Only winners of .15-.18 then engage in bets w. other community citizens. Other citizens can go into debt if they lose a bet, & have little or no "Money".

If a proposed problem is too hard, no one will accept t. challenge; so t. time limit must be t. or t. problem must be made easier (Give a hint!).

.25: .4 Another kind of problem: A math task w. 4 params: 2 Mathns are allowed to select 4 param values for t. other Mathn to solve. First one to solve, wins all of t. money

[One BIG Q: Would this A-life approach be any better than doing about t. same thing directly, w.o. "Alife": i.e. simply try to devise population of competing, cooperating pgms working toward common goal of "Macro TSQ".] We'll try both approaches: See if either can suggest ways to better the other. The A-life ... might

.29 be easy to implement as Trojan Viruses on t. Net of computers. ——→ 169.30

| D. Fogel's checker players are very similar to this idea of competing Mathematicians

30: Spec 167.40 "content" situation has been dealt with to pretty much obtain a pd. of what (t. generator of t. cand.) does next. It could start a new sub tree: perhaps "borrow" a subtree from part of t. corpus; perhaps present as final output, t. output of one of its "sub trees".

.33

167.375 – 168.33 might be regarded as "very close to" or "suggesting a generalization of" Koza's Crossover Scheme.

T. freq. stuff will result in some Guiding Pd" update models that are very expensive! So what we do is usually use t. fast update models, but ever so often, use t. expensive models.

— The frequency of updating is known by eq [124.33]. We can have update

7/4/04
4TM

TSQ .10

.02 (166.10) (Spec ?): Note that PPM usually uses only a short context for prediction — not the entire branch. This is at once better than & worse than Koza. "Better than" because it often detects common/small functions. "Worse than" because while it can detect large common trees, it usually assigns a rather low pc. to them — perhaps lower than they should get. On the other hand Koza may assign too hi pc to them, or just assign low/ish pc's

.05 to common short functions → .18

.10 [SN] TSQ design: Start out w. TSQ in English: but be sure examples are "well defined QA problems: E.g. I could do the HNL of SAARB. Keep operations TM in hi level English, and Express Heurs in English. When the TSQ seems adequate in English then try to formalize it — but do so in a way that enables the heurs to work. "English" gives a "fuzzy" desc of the true process & enables us to "Turn the fuzz" (≡ Make it exact) in various ways.

.18: 166.10 (167.05) [SN] In PPM used for Guiding PD in Lsrch: Note that PPM uses contexts of various lengths for predn. — So if a subfunction is represented as a sub-string,

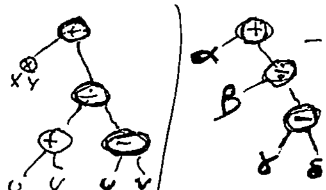
.20 PPM could "recognize it" as a useful predictive context.

Trouble is, all (or perhaps not even most) functions in an empirical Cand, will not be recognizable as substrings: The Tree representation seems much better (in some respects)

If might be that what we want to recognize is say occurrences of args in "exps" in them.

Say A ⋯ Z are "items": Then AXYBEZWZD AUBEXXD are "similar" since they have A, B, D occurring that order; This sort of "matching" is done in DNA analysis. → see 149.25 on Ming Li's ppm to do things like that

What I have in mind: $\underset{A}{\text{Sum}}(\text{Mul } x, y)\ \underset{B}{\text{Div}}(\text{Sum } u, v),\ \underset{C}{\text{Sub}}(u, v)$

30 
— Is a funct of 4 params: α, β, γ, δ.

→ [SN] Perhaps modify notation (Fwd to sa Polish) so that it's more likely to discover subfunctions.

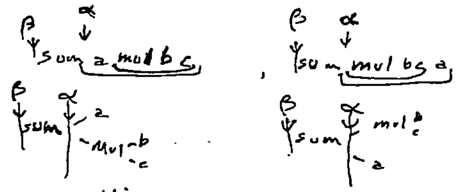
The use of Trees to describe functions is one kind of "Notation", But we have to map it into linear

.37 strings before we can use PPM (perhaps not so! .375) Say we are generating a Cand. & at that point, We have generated one or more subtrees. We can look at corpus for previous occurrences of one or more of those subtrees or of subtrees "close" to them. From the way this (Spec 168.30) →

4TM

5-6

10:165.40 :



context runs left to rt. (as opposed to English context rt to left)
context to rt of α is a mul bc
or mult bc a.

So putting context to left of a function name does not ensure one defines a given "Branch".

In RPN while contexts don't usually represent nodes, they are certainly productive. a mul b c must always be preceded by a binary funct or a arg that represents a quantity (assuming ternary functs are possible)

> 617 642 6657 — is T Mobile no. of this cell phone?

It is of interest to see whether LISP code has different PPM compression backward vs. forward. → 167.18 – 167.00 spec

7/4/04

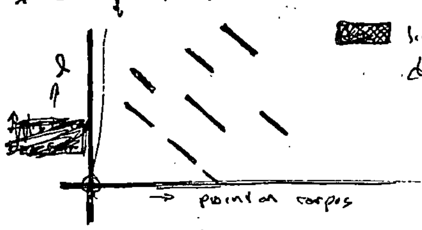
SN In studying the PPM, BW x fn! we are interested in "factoring" contexts:
One approach! write out corpus! in a line, above each symbol in a vertical column, write the case count of that context of length ℓ; ℓ lines above the corpus. From this, it should be possible to find factors.

Today: 231, 300, 513, 15 over "Today"; or "write"; t o d a y)) s with 513, 300, 231, 72

The profile 513, 300, 231, 72 ... may contain info about factoring. If the numbers decrease very slowly or remain constant — they might/would indicate an important new. Sharp drops would indicate new boundaries.
Perhaps we want something like the "derivative" or "difference" function.

Or perhaps $\Delta_k = f(n) - f(n-k)$.

We need run these density profiles out a distance out of only 10 or 20 tokens at most — i.e. the expected length of a "word" (≡ "n-gram").
Would it be enough to just look at the first 10 or 20 values of the profile, to find "new" candidates?
What about relations betw. adjacent "profiles"? We have this 2 dimensional array of numbers — Perhaps we can "spot" words by looking at this 2 dim. array! Perhaps use a "3D portrait." "News" would be ≈ constant line segments of slope −45°.



lines drawn are lines of ≈ constant case counts. They have sharp drops in the +45° direction & are slowly ↓ in the −45° direction.
We can devise 2 dim kernels that could spot words.
It may be poss. w. 1 dim kernels in the +45° direction
a/o " " " −45° " !

So we go thru corpus using forgs to suggest words. We put the best word cands in lexical order & parse the corpus w. these words (+ the alphabet), using a finite "look ahead" parsing routine. & we rewrite the corpus using our word list as new alphabet symbols. The new corpus is then subject to word suggest of .15 — .20 to obtain new words. We loop this until we get no more compression.

→ 198.30 – 199.01

7/3/04

4TM **Ocham's Razor .17** → v.a explain? ⟨Also note 170.23⟩

.00 ! 164.40 body distorts the meaning of fit of hypoth. to test data. In using U.D. This problem does not occur. After we see t. data, we can make any hypoth we want. If it is A.H. it will automatically get small wt.

T. Argt. of 164.30 ff is an impt. part of t. Q: "Why is the Incomputability of D.U. irrelevant to its use in practical induction.?"

.06 It is still poss. to look at a test set, propose a single 'A.H. hypoth., & stop: This Rule seems like a stupid way to do predn., it does seem legit. Is there any such method or general principal, that prevents use of .06? Well there are t. set of k A.H. codes, that each predict a different one of t. k difnt alphabet symbols (w. = pc). Also t. Bern code. A codes should at least do as well as Rules.

.10 — There does seem to be a problem here, in that one could look at t. test set, & devise a seq. of codes that would bias t. proby estimate considerably.

.12 → L srch using an apriori chosen UMC (universal Machine) is a specific unbiased way to do predn using U.D. ⟨192.30⟩

.13 Another assoc. idea: People regard t. choice of UMC as arbitrary, so changes in min code length of a corpus of 10 or 20 bits/bytes does not mean much ... "we can get that with change of UMC choice". In fact one can't change Umc so easily. It represents our apriori info. — changing

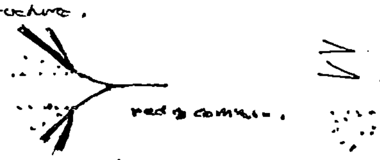
.16 UMC may make Ocham's razor unusable (illegal — since all info is no longer in a priori)

.17 Re: Occ raz: Use convergence thm: T. shorter a code is, t. smaller is rms error wrt generator of data. $\frac{P_T(\text{corpus})}{P_{true}(\text{corpus})}$ (via base code found) → But what about t. future?
.18

.0 : 164.15: In discussing (Goza's Lisp Trees v.s. PPM: PPM is usually described in terms of "context trees". Can we somehow use this to describe "(Program ≡ function) trees" as well? One trouble is that I'm not really familiar w. t. Trie concept. But just as "linear" contexts can be put on a tree, Branches can be put into t. same kind of tree structure.

Superficially, its not clear how to do this!

Say lisp tree common to 2 psms



Another tack! Consider a PPM (tree): each node in t. tree can be considered to be a "context" — (which is t. branch growing out of that node): Hvr. 2 contexts can "match" to some extent, but t. non matching parts can have difnt sub-branches, — so its much more complex than linear contexts. A big Q: A matches B to some extent, A matches C to some extent. If A B & C are Branch contexts, its hard to say whether B or C is a stronger match (as opposed to linear context, where a match of first 5 tokens is better than match of only 4 tokens.). In branches we can also count no. of matching tokens, but t. simple number of matches is not a compelling "closeness" criterion.

Another big difference is that linear context has a lexical order, because of lex order of tokens. But if a bunch of trees match in k (tot) tokens, there is no obvious way to order that bunch of trees. This lack of order makes it hard to "sort" t. tree contexts.

Lets look at Lisp Expressns: (RPN)

7/2/04 4TM

Trg set/Test set problems how Universal D.f. solves it. :30 ff

{16:53 β

13.52P 398V
14.03 : 4.00V — 3am
14:04 : 4.01V
14.06 : 4.01
14.09 : 4.01
14.16 : 4.02V 61.8 ma
14.16 4.02V
14.19P 4.06V ! (Voltage unstable — contacts resistance)
14:39P 4.09V
14:48P 4.01V ! — for restart Error.
14.49P 4.08

00:163.40 : on t. problem of a+a+a ... K times being encoded to Lsrch many many ways, so a stop t. pc of t. next "+a" int. sequence.

However, an "experienced" TM will have found, empirically, that t. prefix "a+a+a..." is likely to be follwd by "+a". Since we are not using t. Univl. d.f. in Lsrch, but rather ≈ "MDL", we don't add all those pc's together (as in ALP).

So maybe PPM would do well here!

.06 (163.40 spec) ! These "rigid strict rules" hvr, are for creating the Guiding P.D. — They are t. "rigid strict rules" that enable PPM. T. grammar generated is then used to create cands in Lsrch order. [We must, hvr, consider one shot learng, otherwise we are too restricted in our srch (too "conservative"). OSL corresponds

.09 .10 to GP picking up a random branch from a random cand — this branch having occurd only once in t. corpus.]

It may be that PPM can do as well as a better than Koza in [illegible] .09 -.10 & in general w. swapping branches (tho PPM doesn't actually swap branches: it "borrows branches".

Koza's "user defined fns": are they expressable in PPM? If Koza actually assigns a token to [illegible], then, Perhaps PPM is "left behind"! .... → (165.20)

.15 Also Koza has a way to do "recursion".....

[It may be possl. to get all of that w. Forth, since forth is universal, but Forth may not assign nearly t. same a priori as Lisp does. (Hvr, we can start w. all Lisp functions in Forth's "dictionary").........

One way to get definitions is by my PSG discy routine. — I also get some recursion by "A→AB"

.20 Hvr, my PSG discy routine is probably very slow relative to PPM. Perhaps start out w. PPM then when we get on a good sloppy part of t. hill, go to PSG discy. When to switch will depend on:

1) Time to execute G fn (≡ "fitness fnct")

2) Expected ↑ in pc penetration due to PSG discy.

.25 3) Look at / eg 124.33 up to for way to do analysis.

.30 HA: (Possibly "Great Breakthru"): In using U. Dist for Prdn: How it avoids "Trg Set, test set" mess.

Say we have some data: Before we see data we use U.D. to make a seq. of models for t. data. When data comes in we start w. first model. If it is "adequate" we stop: if not, we continue using new models, (which always in corresponding to order) [Meaning?]

In t. "Trg set, test set" method we look at Trg set, hypothesize model, then test it on (unseen) Test set. If it is inadequate, too bad, we need new data.

U.D. method enables us to test an arby no. of models using same data — & it uses all data for t. test set, since t. trg. set = ∧(null).

In the 2 data set method, we can hypothesize A.H./models that fit very well, so, if we don't penalize for "model cost" seeing test data excludes A.H., well fitting models. So, having seen data

6/30/04 Phase 1 Induction Methods. 163.19 ff → 164.25 163

4TM 2205 : 6/30/04 β

00:162.02 1. λ ($e^{-\lambda \cdot order}$) param in weighting (use ~~order of~~ G rather than G. (Re using f(G) is usually easier) — value of some f(G)) have to be decided on.

In regular Phase 1 Q.A., we just save the past acceptable solns & use them w = wts as "The Corpus" for updating. For each new problem, we may try for several solns — The they will usually have different G's (= PC's) — but we will be using all solns. in predn. In PPM in Phase 1, we ~~may~~ want more inf. on solns to Recent problems. What we could do is take some fixed population & some G funct., & adjust params of both PPM & conventional GP, & see how good the results of the 2 methods are. We can try this w. various G functs. ~~[redacted]~~ Another way to think about it: for U.D., the test set is all data seen before the corpus (= test set) is seen.

→ [SN] Take all Probs of much interest/ also papers put on CD. The 2-3 the ~~'s~~ of some interest best (Make many Copies. Put in S.D. Box, Give to friends.) its quite possible I'll find a better way to do w PPM. ∴ Do this soon: Do important job, then add ~~progs,~~ papers later. The CD's are "Add-to-able": Perhaps try to do ~~some~~ later version, giving impt. corpus, derivations.

7/1/04 So list the various Progs I want to try PPM on so I can see which forms of it would be adequate.

1) GP , 2) Compression of Eng & of Lisp progs: compare w. other variants of PPM.

3) [redacted] See if [redacted] PPM can compress Σ. 4) Try writing TSQ: use GP w/o PPM to run it.

19 **Q: How Could GP work on Phase 1 problems?** In Phase 1 we try to find short codes for the

20 Corpus. [redacted] The fitness function is a Mesa ∴ No "Hill" just yes/no."

→ Unless we start w. A.H. code w/o promiscuous code & try to improve them.

What kind of fitness funct. did Koza use in his electronic Ckt design? At the start of the program, for a very long time, there would be no identifiable ↑ in "fitness" (i.e. "MESA") In General, it would be instructive to see how GP ~~[redacted]~~ solns. work by converting Mesa (INV) problem's to OZ / Hill climbing probs.

26 In .19 Most (non-universal) methods of induction can be Hill climbers, e.g. finding drivers, & driven consts. Optimizing ~~free~~ params, trying various functional forms (w optimizable params) Universal codes can be found by making trial codes then "corrections" to trial. (MS error predn is a special case of this).

30 In my thinking about ~~[redacted]~~ QA, I tend to think of a search for $O^i(A_i/Q_i)$ in which A_i (guide choice?) ~~[redacted]~~ does not guide the finding of O^i: This is probably a Bad Mistake! Use of correlations, fundamental driven (drivers) etc do involve "looking at the A_i's".

It would be great if I could find a (heuristic) "path" from, say A.H./promisc. codes to a final single short code. My work on PSG Discy ~~has~~ has been in this direction! — Discovery of ngms, then ngmsts, then ~~[redacted]~~ concat rules, — hvr, the forgo stuff is all rather strictly deterministic, non-heuristic srch. — Definite rules for finding these regys. → (164.06 spec)

6/28/04

4TM 161.13 SM

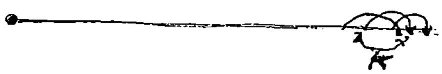
(spec) (160.40) ~~version of the corpus.~~

01: 160.265: TP In the IDSIA report the probability distribution for Lscore is generated by the language AZ....

The C library has a "quicksort"

6/29/04 — Batch → in inner loops" PP302 ff. a very fast radix sort! Memory = 9 bytes per symbol sorted. (V.S. 32 V bytes for 2-3.2004)

04: 138.32 [SN]: On the use of a fast sort that Doesn't Allow Insertion: for a seq. of problems, usually the time involved in solving a single problem will take much more time than resorting the corpus. Actually, if the newly solved problem has k tokens, they can be inserted into the sorted sequence in a time ∝ N, the corpus length. We first find where the k tokens go, then we shift the sorted sequence by k places, working backwards! Until we come to an insertion pt. Then we insert & continue shifting k−1 places till we come to next insertion pt. Loop to α



So it would be feasible to insert each new solution into the sorted list, since this would take much less time than solving the problem.

The foregoing suggests that we start with a few problem corpus & sort the corpus, using any easy way — N^2 sort may be ok. Then after each problem is solved we insert the "k" tokens into the sorted list: No need to "resort".

Another speed up: As we generate cands, we keep track of where in the sorted list each token was found. When it comes time for insertion, we don't have to re-generate that info, so insertion time is essentially time to do N shifts of address of .08 −.10.

A thing that takes much more time in this updating is re computation of the "kernels" for each token. Fortunately, we have a good set of approximations to start with, whenever we do that "Update".

To start off, I could use the common "BZip2" method to calculate probys — but I do still do need access to "innards" of BZip2 before I can use it much.

Actually I'm interested in 2 Q's: ① will my method of prediction using "Kernels" be better than current methods ② How good is PPM in general for PG goal? Perhaps they should be addressed w. separate softwares!

6/30/04 What about applying PPM to GP? How often do I need to update the PPM prediction system? Well, depends on how fast we discover good new cands! — also see result equs on this p. 123.32 (124.33) — depends on how much extra ΔG we expect from newly discovered good cands.

N.B. the device of .08 −.10 can also be used to remove elements from a sorted list.

For GP one might do "Generations" — so that one makes crossovers & new population from the old via PPM: then we update PPM on whole new population at once. Certainly this is a common way to do GP, so we can easily compare it w. the PPM method. A suggested tack: Compare mean ↑ in G per Generation betw. Regular GP & PPM based GP. In both GP's PPM version, there is a biasing toward offsprings from hy G parents. ("Batches", Population "Batches")

In both GP & PPM version, this can be done in various ways.

N.B: Strictly speaking the use of PPM in Phase I (an optz problem) is not exactly the way GP works: — In Phase I, the largest G (highest PC) cands are always retained. In Phase I using PPM, we can update on top G fraction of population, or use whole population with "weighting" toward hy G. The value of α (163.00)

6/27/04

4TM [3M] ...

20: From 4TM 129.20: A criticism is that the drivers & the driven sets are the same. It would be better to have them be separate. We want large σ^2 for driven, but σ^2 of drivers is irrelevant.

If we have several drivers for a given driven, we can study recent configns of drivers (context).

e.g. the confign of drivers on previous day. How to do "Partial Matching" is unclear!

If we have N drivers; each context is a pt. in N space, so we can use a metric (Euclidean or other) to tell when today is ~ to other days in past. Certain of the N components will be more impt.

Then others ... giving the metric. — The this is beginning to look like linear predn!

— Not quite! is more like "Case Based predn" ... we look for situations in past.

Anyway, A BIG Q. is are configns of stock prices good drivers (≡ predictors)?

Since single stocks are not bad drivers, configns can't be worse, but are they really significantly better? (considering SOY diffity!)

→ 171.20

6/26/04 Why Phase 1 w. "reasonable" TSQ "has to work": .00

.00 (Space .158.30) : SN on Adequacy of PPM for Phase 1: W. an "adequate" TSQ, PPM has to work! — (presumably the language used is adequate). While PPM ~~is~~ used for Compression, needs large corpus (≡ large TSQ) it shouldn't need this for Phase 1. — I think I have all of the needed induction mechanisms in Phase 1 — as is, PPM is pretty much a kind of Bernoulli's or "Laplace's rule" type of inducer — which should be adequate for Phase 1 (& perhaps for all induction!). Try to find/reconstruct the SAARB tsq for ANL. See PPM ~~pets~~ about same pc's as ~~everywhere~~ PPM. — Perhaps w.o. using STORAGE instruction! — since PPM is supposed to "simulate" "storage" (or actually Emulate it!) If a conc. seems to have its sub-concs in the tsq & they have become .11

.08 In Para Notes (last 7/100 pp in particular), Realize several impt. ideas on TSQ construction/design. ~~Pear Tryaa~~ Index these remarks & try to summarize most impt. ideas.

ABCdefg ABCDEFghijk $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2}} dx$

11 (.08) → Trnd — yet the composite conc is not learnable in reasonable time, then there is one or more Heuristics missing from the Conc-~~set~~ — — that have to be trnd first: These can be trnd directly (perhaps as hy cc) or w. "hints" to various degrees, ~~or~~ or, the ultimate "hint" is to ~~PPM~~ ~~into~~ ~~have, directly.~~

16: SN On using a 2-3 Tree for PPM, for cand ~~separation~~! All previous cands are represented by strings that start w. the start symbol (S̃). ~~Bef~~ Before we insert a new context (i.e. the front part of a cand) into Σ̃, we change its (S̃) to (N). (N) ~~[…]~~ is a unique symbol that occurs nowhere else, so it always causes mismatch at that pt. ~~After inserting~~ (N)–(S). The ~~forgy~~ trick is to prevent matching from going past S̃, to the previous cand — which is irrelevant.

0: 159.40 : One Limitation of Phase One is that it makes new trial Oʲ's by looking at previously successful Oʲ's and making trials that are "similar" to the successes of the past. Though it does not understand ~~[…]~~ opt. optimization ~~is~~ is what is wanted, ~~[…]~~ it is probably capable of finding reasonably good Oʲ's. Its inaptitude in this area is corrected in Phase 2. ~~[…]~~

.25 Using a corpus of ~~past~~ successful Oʲ's of the past, Phase 1 induces a distribution .255R .35

26 .255 function ~~of~~ over new candidate (trial) Oʲ's .255 → It then uses Lsrch to find cands. of max a posteriori probability. In the ID 5/12 report, the ~~[…]~~ distribution for Lsearch ~~[…]~~ probability Ma is generated by the language AZ (see Appendix A ~~[…]~~, p27 of ID 5/12 report)

In the system to be developed we will be using a PPM-like method to generate new Oʲ's from a corpus of successful Oʲ's of the past.

30 I ~~[…]~~ had a specific modification of PPM in mind, and I wanted to test it on English text (& perhaps Computer ~~pr~~ pgms (Lisp, say)) — The Calgary Corpus has some Lisp! ~~to see if my mods are better~~ to compare my modifications with other variants of PPM. ~~Then or comparable to other modifications of PPM.~~

[Follow w. a detailed explanation of How my version of PPM works]

35 : (.255) (.255) Insert at .255R When an acceptable ~~found~~ solution has been found (The criterion for acceptability can be various: (1) Max Search time is ~~exceeded~~ reached: Pick best set of cands thus far (2) Total ~~prob~~ a posteriori probability of cands is > a certain threshold (3) Trainer may decide when search should terminate. — on "Heuristic Basis") This ~~new candidate~~ solution is added to the corpus of solutions.

Periodically, the ~~[…]~~ probability distribution used in Lsrch is updated using the latest (Space 162.00) — Version of the corpus

: So: Abstract/summary of Review "Started" on 139.00 139.00.

What I want to do is program a prelim version of "Phase 1" (Phase 1 is desc'b in pp. ..... of IDSIA report)

I will be using PPM (or a variant of it) to assign pc's to Tokens in the generation of candidate solutions to problems.

The sequence 139.06 R : then later 158.11 – .30 ... to B discusses why I thought PPM might be very good for this task: how it could do some things that definitions do. — Some things done by definitions that PPM seems unable to do.

SN A (perhaps) different way to write this review: Just write it up in "sequential order" of ideas. Then gradually add comments & refs to comments. As I do this continue making indices of imp't. ideas found in this region of the manuscript.

0) What "Phase I" is: how it just tries to make coarse "i" to previous cards.

e.g. 1) Why PPM would be good for Phase 1: (Comments! (Pro), also definitions of PPM (Notes on this: (00 R)))

2) Ways to realize PPM (for Me..) 2-3 Trees v.s. Quick/Heap Sort (& other sorts in "inner loops")

3) How to apply (PPM) to Phase 1 problems: 3 kinds of L search: ① KT; ② LL; ③ Random. Advantages, disadvantages: why ① was chosen: (K prob ≥ 3: work out proof again).

This Memo will describe/introduce the use of PPM-like induction to realize Phase 1 of the Alpha, AZ program.

We will first describe Phase 1, how it works and how it differs from Phase 2.

The PPM compressor and prediction program then described, and we show how it is to be used in the L search part of Phase 1.

There are several ways to realize PPM and related programs. We will describe some of them and explain why they were chosen.

There are also three commonly used methods of doing L search. We will describe them and explain why we selected a particular one for Phase 1 induction. — The Levin doubling method; the LL search method; Monte Carlo method.

Section: What is Phase 1?

Phase 1 is desc'b in § 7.2, p.24 of IDSIA Report Rev 2.0 Oct 2003 TR IDSIA-16-03

However, instead of the AZ language, we will be using a PPM-like system to obtain a probability distribution over trial solutions to problems by generating its Tokens, one by one.

The problems of Phase 1 are all "Q A" type problems. This is a fairly general (but not completely general) form of induction problem. The main problem in Phase 1 is always to find a function, O^j such that $a_0^j \prod_{i=1}^{n} O^j(A_i|Q_i)$ is as large as possible.

See (Section 1 of IDSIA report: p.7, eqns 3 and 4 for more fuller discussion.) — a_0^j is a priori of O^j

→ 160.20

Right margin: Was unassumed: PPM: 131.05; 130.06; (130.27 R); 133.05 conclusion of problem solution; 132.27ff; 133.02; Also not 131.23

Vertical margin notes: Actually, we can still use 4. AZ language, but we will use PPM to assign pc's to successive symbols in the cards. We use eq. 37 w. n>1 to solve seq. prediction problems, if we set Q_i to Λ (null) and O^j's that generate A_i and need not stop after generating A_i.

6/22/04

4TM

Mur251 usr (pwr) LoveLO all lc. loveLo } Murray at Yahoo.

That PPM May be Adequate for Phase 1 & Beyond! .11 - .30

00:157.40 :::::: I want to see how t. forgp. ideas (13 here) related to actual TM problems in which PPM is normally usd.

Well, we use PPM to get pd for next token of a string; based on either statistics of a long string or of an unordered bunch of shorter strings (say strings that represent relatively successful cands for solving some problem. So consider we are working on a QA problem in Phase 1!

We have this historical $[O^i]_{i|N}$ corpus: PPM gives us a distribn based on "recent past" for each token freq. during cand generation.

In one sense, the PPM is "Universal": for a suitable corpus (≡ TSQ) we can get any pd. we like; in t. sense that for each of a certain set of "contexts" (≡ recent past data strings) we can have any pd we like for t. next token. PPM is prob'y "consi", for any of a certain large

10 set of stochastic data generators — Maybe Finite State Grammars.

11 6/23/04 A Major Q.: If we put in a part of a TSQ that is supposed to embody a certain regy that we

12 want TM to lrn: Will PPM enable that lrng w. reasonable SSZ & cc?

N.B. For certain theors, the "corpus" must include "traces" of previous problem solns. — not just t. past probs & their solns. How PPM would do this: I haven't worked out yet.

15 Re: .11-.12 It would seem that answer must be "Yes"; If t. component concs. have acceptable

16 pc's then t. pc of t. conc to be discovered will be "acceptable".

17 T. Trouble w. .15-.16 is that in PPM all pc's of tokens or concs are conditional on previous token seq. in cand. In conc. nets (as in Sol 89) t. pc's of concs were "unconditional". Using PPM can vastly ↑ (or ↓) those pc's. I guess t. conc net "idea really is incomplete — it's t.

20 "outline" of an idea; but "context" is very impt. in assigning pc's to tokens.

(Which was a Serious Error in Sol 89. I didn't then realize importance of "context")

From t. forgo. (≈ .11ff), my impression is that PPM may be good enuf to (solve very hard problems) discover very diff't concepts) in Phase 1. That Phase 2 will be ① a way to get TM to "understand" what "opt. means". ② Get beyond Lsrch for a "lovely" (no "cong's factor") optimum srch.

.051792

Hvr, using PPM, it will not be obvious that a certain cand has hypcs & pc (to t. trainer) will have to be evaluated by t. PPM. Writing TSQ's will be very empirical & intuitive.

Still, PPM seems to be capable of recognizing only 1 kind of "context". E.g. t. source of a problem is an impt kind of context, yet it is invisible to PPM. We could

30 indicate t. source by an "index" on t. problem. How could PPM deal w. indices of this sort? → {160.00

6/21/04

4TM

Note 166.11ff on How to use t. BW xform to discover "words" in a corpus.

:156.90 run into trouble when we define > 1 ngmst, because of overlap of ngmsts — so we have several possible parsing methods for t. corpus. (In recent AZ141 we had parsing problems as well … we wanted to [illegible] s. ↑ in pc due to alternative parsings) in Ru[illegible] — as well as recent ideas on ngmst definitions, we do as Wolff! reparse corpus after every or every several new definitions.

157.33 ff tells how to generate "token sets" — So we first have to have defined some tokens (which, typically, are ngams). We can also use t. BW xfm to get freqs of various ngams, quickly, so we can use AZ141 to define them.

.06 AZ141 Def — I'm using AZ141 in t. sense of the way to assign PC's of single ngms, that I desrbd in my long letter to Gerry Wolff.

Is it possl. to use t. ideas of 156.33 — 157.08 to ① define ngms of longer length incrementally (as envisioned in t. original AZ141 report) plus define ngmsts. ? If we could do that we'd be well on way to CFG discy, or even w.o. CFG discy: Ngmsts are good enuf to do rather good predn.

T. main "New" ideas ① use of BW xform to speed up such for tokens and also token sets. ② frequent reparsing to get at least one good parse. ③ T. idea of hillclimbing to incremental changes in (word to ken defns) & (ngmst to token defns).

An additional theor: BW xfm enables us to find ngms of hy freq. that have a very idiosyncratic set of tokens following them. ngms of this sort are a cand. for a ngmst defn. Also t. d.f. of tokens following such an ngmst is also a "Symbol set" cand.

20 I've written various times on ways to get ngmst cands for FSG discy: these methods would be speeded up by using t. BW xform.

.23

.24 : 156.29: T. reason this is very impt: That I may find a way to use/modify PPM so its a much more complete/universal system. <Also, I will understand better what's going on!>.

Say, t. operator α reverses order of [illegible] last 2 tokens, so that if we use RPN;

a, b, α, F [F has 2 args; α has 2 args.] = b, a, F.

= (a, b, α), F = (b, a) F.   So say we had t. sequence a, b & we want TM

30 to discover that a, G (RPN) is t.g. — even when (Drop that erased line)

Say PPM gives a certain PD, based on observed correlns betw certain "contexts" (= ngms in this case) & t. following tokens. For each context C_i, PPM gives a d.f. on tokens = $P(T_k / C_i)$ ← : Say G is a function on t. poss C_i, that can permute them — thus changing t. P.D. in this arby. way — i.e. we get $P(T_k \mid C_i, G)$. (RPN)

Which is a quite diffrnt d.f. on the T_k than → given by ?

We may think of G as a "P.d. modifyer"! a common example is a Tm: P.D. on input → P.D. on output.

(See 158.00) →

4TM

PSG DISCY .30 to 157.23 Cat least — William O. Felsman. R?

10: : [SN] On Phase 1, PPM, Phase 2: .... Intuitively, it would seem that kinds or theory, cares, should be learnable in phase 1: If they are expressible in t. (universal) lang that Phase 1 uses, then they are learnable ... via PPM.

On t. other hand, PPM can only observe recent context, clearly not all kinds of "context". — Which is the main reason why we have to go to Phase 2 for very smart behavior.

05 [Also, phase 1 seems unable to really understand what optimization means]

So .00 – .05 suggests some confusion in what Phase 1 w. PPM can & cannot do. & whether Phase 2 is nccy : or perhaps another Q is — Is phase 1 nccy? { Couldn't I just devise an (universal extensible) grammar for PSM's & try to correlate/relate problem being to t. set of PSM's ?

10 (This last is a kind of QA induction .... so ≈ phase 1 for first approxn.)]

To argue against PPM: Consider a set of probs that are similar (in t. sense of being solvable by ~ or t. same algm) — say these probs are of t. form $x_i K y_i$ where K is common to all of t. problems, but x_i & y_i may vary between problems. As is, PPM couldn't notice t. "similarity" K, because of t. barriers to "adjacency" x_i, y_i. I.e. In the problem, soln, pair ($x_i K y_i$; soln) T. "context" K has no immediate following effect of by likelihood.

K and soln are "linked", but not adjacent.

Note. If we could somehow generate t. ngrmst x_i & y_i then I think they would Solve t. Difty

20 I certainly had been thinking of ngmsts (≥ dlays → slays → pd's) — But I don't think I ever found a good way to extend PPM to ngmsts or pd's or dlays or slays. → See .30ff on latest "Best Part".

PPM is already a kind of ngmst — a sort of more like a slay, because t. elements of shorter length have less/more wt. One way I considered: Consider all contexts (preceding) t. ngm α.

This is a kind of slay. We can context (cont. right) a ngmst. of variable strings, so we get t. ngmst. ngmst, *(≈ angles) — which means ngmst can be followed by some random finite string. & this regular Ngmst can be followed by some kind predn, with a distance that is a P.D.

Hvr, .00 – .10 is t. Main Argument : I really have to Get it straightened out !

Just "Thinking Out loud": We can perhaps simulate a univl. D.f. of my PPM if we consider

29 a PPM weighted seqs of universal instructions. → 157.24

30 [FN] As SSZ → see perhaps PPM "must" (?) approach t. universal D.f. i.e. in this sense it would be "Consi".

I think this is what I've been mulling about! PPM defects a certain kind of recy on "passive" symbols.

Hvr, "active" symbols, make it possl. for certain symbols to modify t. meaning of previous symbols.

33 Say "α" means "erase t. symbols first". We can code t. corpus using α followed by selection of which of t. 4 it is (pc will be α freq of that symbol within α's occurrences).

So we can code t. corpus & get its pc by using t. deltn, α. Next we can hill climb on additions & deletions of symbols to α. This mite be done rapidly if we first

BW xform t. corpus ! — since then it's easy to find all occurrences of 1, r, s, x.

We can evaluate corpus pc's via AZ 141 formula. While this speeds up AZ 141, we still 157.00

Side box:

Ind. concepts: 145.17 PB → PBCC Not solved. 145.16, 145.18, 146.17 see also

Speeds to 2-3 Trees Many stages 147.02 – .14 Alife 147.28ff ANN (5)

TSQ 142.28, 143.04, .12, 152.30 good! 153.38 – 154.12 137.26

Low prob Coding 154.30

IND Compound Analy 155.00

4TM

→→ Several Tutorial papers on web: also see D: PSI 6/17/04 (P28)

00 : [PN] ICA: Indip Component Analysis: May be of Much interest. Is normally "Linear". Joergen Schmidhuber (& perhaps Hochs..) did "non-linear" ICA w. ANN (so normal ICA is "Linear")
Also look at J. & H's paper on "Flat Minima" for ANN. ... They got some reasonable SM results. Look at Juergen & Hochreiter's web sites for papers. There is a report on "Flat Minima" in pseudo code! I don't know which one, hvr.
— maybe ask J. : It's like a souped up "Factor Analysis"? But like Zero code detection?

06 : 15440: Each set of N pts is a pt in N space: we could normz. & power & reduce no. of dims.
The th. no. of chars in "alphabet" is large, & still amenable to PPM. We may have to do modified "escape" since Librarying that we will often observe chars that have not been seen before. We do, hvr, have a (euclidean) metric betw chars, so if a char hasn't ever occurred,
10 a "near by" version has. A method of searching for "near by" strings of chars, ... unclear how to do this.
One way to deal w. problem is to quantize signal so as to minz total be of signal.
We can do this repeatedly for dif't "error" levels (rms error) for the LPC code.
There is of course an optimum mix of (LPC code + code for error) i.e. MDL —
This may be a way to get "most accurate" LPC code — which might be good
to Use PPM

8 [SN] There are ≳ 3 Major problems in Getting [GP] to be "practical" (Phase 1)
1) More efficient search over popln for cands (PPM seems like a good try") (This is a usual induction problem & an OZ prob. (in advanced TM — perhaps in phase 2?).
20 2) What to use as initial population
3) How to recognize that a given problem is similar to an older "solved" problem.
4) To translate a formal "welldefined problem" into an initial population & a problem soln lang (This could be partly from "Macroscopic Content") a suitable (& perhaps a suitable mut/cross algms) (if PPM is not usable.)

(2,3,4) are closely related (maybe identical).
Anyway, after TM recognizes that the new problem is ~ to degrees a_1, a_2, a_3, resp. to past probs, $prob_{1,2,3}$ — we can look at the populations of these 3 problems at various stages of completion (call that test ... a no. betw 0 & 1 ≡ α).
30 So maybe mix the 3 populations w. $\alpha_i = a_i|_{1/3}^{1}$. Actually, we have 2 params for each of the 3 populations: a_i (closeness of probs) & its corresp α_i (closeness of population to soln, final). We must somehow mix the 3 populations & the 6 corresponding params to get an initial population for the "new" problem.

The Initial xform of the input problem is into 1) A lang for expressing soln
2) an initial population 3) possibly a mut/cross or PPM or some other way to generate Cands.
More Generally: how to xfm input prob descn into an appropriate PSM for it.

6/14/04
4 TM

Very General Approach to "Phase 1" : (00) (≐ 153.38–.40)
LINEAR Predictive Coding (.30)

Spec

00 :: 153.40 : Given PPM, a TSQ : a^{succesive} CJ's for Ones : we have the hillclimbing problem of reducing the CJs. The kinds of possl. "Moves" in the climb are like 153.38–.40 — They are ways to train the trainer (among other things).

Actually, the Goal of our "Hill" is the Σ CJs for the entire sequence. Say we keep final problem in the TSQ constant. We could change the seq. of probs a/o Modify PPM or use a method of induction much diffrnt from PPM (say GP).

.06 If I use PPM, I may want to "BOOT" TM with a large set of "free" solns. to a large set of probs. This is because PPM may not be able to do much w.o. a large SSZ.

→ If so perhaps it will always learn very slowly: needing many examples of a conc. before it can usefully use it in prod. Corresponding to J's OOPS results, If I wanted TM to lrn recursive functs, I'd put lots of them in the "Boot".

.10 Re .06 ¶ : If PPM is slow, — well it's probly still much faster than normal GA.

The Q is : would it be as "Creative" as GA, or would it be too "Elitist"?

.12 Also I suspect that local Extrema using PPM for GP probs will not be a problem .... but we shall see! → (.22)

13 : 139.40 [SN] I had idea of applying PPM to the BW xfmn. Since the BW xfmd corpus has same info (except for $\frac{1}{N}$ factor, N = length of corpus), any compressing we got would be useful (except for additional $\frac{1}{N}$ factor!). Well, I believe that successive use of WB xfm does not add extra $\frac{1}{N}$ factors! They merely permute the corpus: so we only have to use the $\frac{1}{N}$ factor once. If this is true, it makes it much more likely that the subseq. WB xfms will give some extra compression. (Try to find previous ref.!) Sec and 154.40

If the first BW xfm shifts corpus by α_1 & the second BW xfm shifts it by α_2, then to get the original corpus, we need only shift $(\alpha_1+\alpha_2) \bmod N$ — which is of PC $\frac{1}{N}$.

.20 I'm still not really sure of this, hwr!

.22 (.12) → PPM didn't do well(?) w. short corpi : Perhaps for a long "time" (many probs in TSQ) the bc per problem will continue to ↑. — Until it has a large enuf corpus; then the bc per problem will become more "constant".

.23 I also considered (.06) of starting TM w. a bunch of prob/soln. pairs so PPM would begin w. a reasonably useful set of prefixes, so the "flattening out of .22–.23 would occur sooner in the TSQ.

But, I think the main idea in TSQ writing, is to really know ~~reasonable heurs for each of problems~~. Ideally, these heurs should be lrnd by TM, if not directly, use as few "hints" as possl. to get TM to acquire the heurs w. acceptable cc. We necessarily need TM to be able to discover heurs of all kinds.

.30 : [SN] Linear Predictive Coding is "similar" to PPM in that both xfm original signal into a signal of "narrow band". Also, both use "recent context". I Guess LPC is in 2 parts (1) The set of slowly changing coeffs (2) the error in prediction. For lossy compression, the error signal may be omitted.

LPC as is would probly be poor for speaker ID. Hvr., say we use 4 coeffs: This is a pt in 4 space. These "pts in a space" would be characteristic of a speaker & could be effectively coded via PPM. So, each of the 4 coeffs is xmtd digitally w. its own precision level.

It may be possl. to use PPM on that set of "characters". Each point in 4 space is a symbol in the language. The density profile of these "Chars" in 4 space could be one characteristic of a speaker. — But sequences of characters (via PPM) would also be useful.

(discernible w. PPM)

155.00

TSQ.12

Charles Bloom has written about ways to Improve PPM... & has implemented Many of them. I have a relevant paper By him.

"2 kinds of PD's": explained in 3TM 424.06; 3TM 394.22 Wiseway version of problem

PD1 to define Goodness QATM; PD2 "guide search"

Spec

00: 152.40: Making Conc.nets for solns w. "Logical reasoning": Logical reasoning "maps"/xforms a Given problem into a problem in Logic, Then solves it as a logical problem.

So Given problem: To try to map it into a logical problem is one possy. (We want to use Context to make this more likely (if t. context is appropriate). If this is successful, we continue w. that soln. If not, it doesn't appear in t. conc-net. (only successful/useful operations occur in t. conc. net).

TSQ v.g. 3TM 425.00 .. 419.29 .. 403.00; " 402.12 on various kinds of "context". 3TM 390.08 on "Context"

If ≈ PPM is "inadequate" for Phase 1, this means one or more of foll'g:

0) also Note 158.17!

1) SSZ is too small in TSQ ∴ pc's are too small.

2) Conc net has to be "denser" more Skinner-like learning. CJS better. Concs has to be

3) We need a faster CPU a/o wait until faster ones are available.

4) Similar to 2) We need a "Better" TSQ.

5) Some time ago, I wrote about "TSQ" Repair → find it. → TSQ repair is 15.20. It was around No! (IDs 3TM 405 or ~411 421. I scanned 3TM 386–452 Couldn't find "TSQ repair")

12 - 152.40 I've written lots about TSQ writing: One v.g. idea was to write "rough & dirty" TSQ's by writing an ordered list of prob/solns in actual ppm code.

Then look at CJS's. Try to Repair by adding problems, modifying TSQ, improving Context dependence of tokens. (150.10 ff) When I finally do get to TSQ writing, perhaps read & review last few yrs of TM writings & try to summarize impt. ideas.

One Complaint about d induction TSQ's is that they are not readily prepared to Sinduction problems. Hvr., Re: d induction, If seems to be easy to find reasonable probs (e.g. Iris, the Mpg data, ect) for it, & it would teach me more about TSQ Writing & conc-net construction.

[SN] On finding Shortcuts to Solve QATM: One easy way to get a pd: have O'(Q_i) give 1 (or a few) (or many) outputs! Then we have these outputs as "centers" of ∈ (vectors), w. a "distance function" from cluster center that determines a pc. We may have to renormalize. The induction function itself could be a clustering Algm in which, in t. past, the algm has been successful in finding sets of cluster centers & assoc distance functions to desrb solns of particular QA problems. — i.e. input = Q_i, output = set of centers, & assoc dist. funct. The distance funct can allow all Ai's to have pc > 0; — which is very desirable! (This often pc assignment methods would often give pc = 0, was a profound criticism of them).

[SN] On "2 kinds of P.D.": PD1 Defines t. Goodness for QATM. PD2 is a function of how many QAi's we've done. It's an attempt to get t. PD assoc. w. t. Corpus. Q1, Q2 ... Qn. Essentially a SUMAC. In addition to that Corpus, if QA's & any External "Contextual" info should be included. It's TM's PD. for On+1 in view of all of t. info that it's ever been given. With corpus Q1, ... Qn; PPM could give a PD on [illegible] On+1

.38 — T. Q's: ① would it have usable CJS? ② Could we repair t. TSQ? to give acceptable CJS?

.39 Could we "repair" PPM to give usable CJS? The TSQ may be beyond Phase 1. Use an entirely

.40 different TSQ! — See sec. on "TSQ repair";

(Spec 154.00) →

4TM TSQ: .30 Impl

0: (Spec 150.40)! To start w. Struct Gov: We have a set of N problems; assuming it O_i solves m out of N, it gets $pc = \frac{m}{N}$ — which is a somewhat useful hill (Tho it considers all probs to be of = value). In a next step, it may try to modify recognize probs where O_i didn't work & find corrective strns. Or find corrective strns on O_i that work for all (or more) problems high below. This is perhaps same of N to what I considered in the latest version of IDSIA Report: early chapter on QA lrng.

Hm, if we use the current QA Gov; Then any unsolved problem gives $pc = 0$ or ($\ln Gov = -\infty$). If we assign $pc > \phi$ to unsolved probs, we can continue to hill climb on other problems.

But less "interesting & sharp"! I want to get to the TSQ writing part as soon as possl. — since this seems like the most seriously diff part of TM design. The early TSQs can be practically anything. They are mainly designed to teach me how to write TSQ's.

0 OK: Back to 139:00 outline of review.

I may want this typed up to refer to in future — I.e. IDSIA report.

So start w. dern of Phase 1 of TM & also Phase 2: How they differ! Give (narrow) refs in IDSIA Report.

The present most direct goal is to use PPM methods to assign pc's of tokens in Phase 1, L srch. for QA problems (both d.fncts & s.fncts).

.10: 150.10 SN (= NMTM) for S Induction: At first use Barron inductively (freq concept).

So induction becomes problem of finding (easily defined, hyp pc) sub seqs (not nec'ly contiguous sections of corpus) & "Alphabets" of tokens for these sub seqs: so one can do Barron induction,

I.e. we want to be able to "partition" the corpus into more easily codable/predictable parts.

.20: 150.26 → The idea of set of (PPM type) concepts is of interest. It increases the size of the data for predn. One way to get such Nzansts: Consider the set of all (previous) contexts that were followed by the token "c". Put them in (backward) lex order. We may be able to find Nclusters as "clusters" of prefixes.

.30: **TSQ**: It would seem that writing TSQ's would, by far, be the main problem in TM. (This doesn't necessarily make it a diff problem, hvr.!). One writes the TSQ & its Concept net. From this, it should be possl. to estimate CJS's of solns to problems. Using more sophisticated Phase 1, Lsrch "Guiding Pd" will perhaps signify ↑ pc's of tokens that desc. solns. — But after the TSQ is written we can then look for contexts (150.10–.26; 152.20 ff) or ... of context that would be helpful for the TSQ solns expected by Trainer. This should tell us how to augment/modify PPM to ↑ pc's of solns.

Spec. → 153.00; .12

6·10·04

4 PM

ANN : .00 ff is for usual feedforward nets: No real soln. .20 ff is a soln using cross validation. That looks good.
– .09 | .10 – .15 is a soln. to general n-l. curve fitting – that can be used in ANN.

: ANN! our approach to analysis. (1) Conventional feedforward, reinforcement.

w. Cross Validation: Trg. set vs. test set.

Random initializations of net. Progress of wt. adjustment on trg set until fit on test set is best. done for many random initial [redacted] izations. Prediction based on some (wtd?) mean of "best" ~~various nets~~ various /codes

Analysis: Each randomly chosen code is of same prob. The trg set changes the initial params into a single code (by choosing "Best"; assoc w. each random initialzn, is a set of predn. stats to be used on test set. Because of ~~random~~ random initialzn, each ~~set of~~ set of stats is of equal wt.

It is not clear how to use the trg. for predn. a/o how good the predn. would be. → (.20)

Alt. start: ~~Use~~ a standard ANN w. small no. of params. pick random params. Use ~~[illegible]~~ N.L. optzn to get peak pt w.r.t. "Data set" (= test + trg. sets) on the

2 – (43.11) This case (.10 ff) is essentially n.l. curve fitting: Using random initialization, we optimize to peaks, & get hessian (= width) of each peak. During optzn, we get idea of how large the "goodness" is

3

4. of a random pt. in param space. For predn, we use a wtd mean of predns at all of the peaks. Wt. is ~~[illegible]~~ ≈ ht. of peak mult by its Hessian. We can get total hypervol. (= Σ Hessians) of the these peaks & get the mean (non peak) "background" from (.13 – .14). I think this will give an "unbiased predn" in some sense.

T. trgg. does not solve problem of "How many params" — which can be solved 2 ways:

1) Somehow use past data to estimate the increase in precision assoc. w. the no. of params – very General, e.g.

15 2) use $\sigma^2 \to \sigma^2 \frac{N+k}{N-k}$ formula from linear predn; I'm not sure of "Proofs" for > 1 dim)

(.10) → Using only trg. set info: Assoc w. each initial pt. will be a number, N_i, telling how many trials to be made to get to "Best" pt. This no. will be "$\pm\Delta_i$" i.e. say a second order peak so we don't have to know N_i exactly. Maybe PC ≈ $\frac{\Delta_i}{N_i}$

So ~~[illegible]~~ we could do wtd Σ of initializations w. wts ≈ $\frac{\Delta i}{N_i}$.

I.e. Predictions are ↑

Each possl. division of the Data set into "trg." & "test" will give a different amt of ~~expected~~ variance ~~[illegible]~~ in the predn. — (depending Mainly on sizes of the 2 sets). So we select a size ratio to minimize expected var. of predn.

10 : 139.00 is start of review: Some impt ideas to include "redundancy"
3TM 420.20 starts work on the "redundancy problem"

02: 139.06 R T. Q of to what extent PPM is as good as (or better than) t. use of definitions. (Ref: 139.06R)

At that point, I felt that definitions summarized lots t. use of them to "rewrite" a corpus, saved much time in looking for hyer order regularities.

It may be that PPM is as good as or better than defns, if one only considers "one layer": But defns of ngms (= "words") make defns of NGMs possl... which is an impt "Hyer order Concept".

3TM 403.24 discusses t. 2 PD's": I don't remember what they were — tho I thot it was very impt at that time!
2 possy: 1) In Lsrch for Phrasal induction: T. guiding Pd is usually a fairly simple Pd, like AZ or oops... very Bernoulli'ish: BZip2 is a more sophisticated possy.
2) Could be t. "True" (incomputable) PD implied by some UMC or equivalent.

0 Another thing I discussed much in t. past was "What is Context" I think t. final conclusion was, that (in essence): It amounts to "How is t. Pd of t. next symbol (or t. Pd of t. near (or distant future), dependent on t. past of corpus (or any other Aux Info)"

PPM uses a simple freq. model of proby — w. some needed refinements for "symbols that have never occured before (escapes)". But, in general, t. Q of "What is Context" is t. same as t. general prediction problem.

How can we Refine PPM's concept of "Context"? — using ALP, say.
AZ-14 is a sort of way to do this.. Using definitions from Ngusts. Expanding into CFG, ect.
I don't have a Grasp of how to look at t. problem in a good, General way.
PPM: "plays Corpus (reversibly) & $\hat{L}$". Can I think of other, better, ways to predict on $\hat{L}$? I've written some on this last.

6.12.04 Originally, t. idea of "Context" being impt arose when I was using simply t. past freq. of tokens to get pc's for t. next token. "Context" suggested ways to get better pc's for tokens.

0 But in general, ALP gives t. best ways to get pc's of tokens & is t. most general way to get pc's.
"Recent NGM" is t. most restricted idea of context... this is used in PPM.
I think that this is modified to consider ① size of each context & ② how often it gave rite pred. (i.e. how sharp its implied P.D. was) — also was it "determinist" (= only one continuation ever observed). For some ways of using contexts, they observed that using longer contexts ↑ entropy/symbol. This must be because they were not using rite algm for evaluating effects of long contexts. I treated one aspect of this in my analysis of

.26 "One Shot Freq" in AZ 14.x (see letter to Wolff). ⟶ Another tack is 152.20

SN Most recently, I've been considering having TM start of on MTM problems, then
1.4 go to what seems harder: NMTM. However, maybe better to start w. (stochastic) functions
30 (= MTM) at t. beginning, because one is almost immediately "on t hill".

In many optzn probs, getting on a reasonably sloped part of t. hill, is t. initial & often very difft. problem.

We could use a MTM tsq but a start for evaluation — so TM wouldn't have to get all problems rite to get a good score (actually any score at all!). In starting w. a true MTM Govc, I had considered t. problem of moving onto t. Stochastic Govc. I don't know if I ever found a good way to do this!

Would starting w. Stochastic Govc "immediately" be better? — or would I have some difficulty transitioning to true NMTM problems.

→ (see 152.00, 152.16)

6·7·04
4PM

0 (Spec 148.17) cont.: I have been considering ~~competition~~ among cands: Could they not cooperate (also/instead)? be designed to
T. advantage of Competition is that this furnishes a mint TSQ. Could we obtain such a thing w. Cooperation? (Also Note 148.30)

So: Q's: ① What are good algms for competition or Cooperation in a Community of Cands?
② Could such a community "live" in t. environment as "worms"? If so, how would they deal w. less (benign) (benevolent) worms/viruses/spyware/trojans/?

07 As is, Ray's "Tierra" cands could, perhaps, if suitably designed, live in any computer as a "virus" & have offspring, that it would send out over t. net. Here t. goal of t. system would be that of "Real Life" — i.e. no goal at all — just reproduction as a "sort of" Goal. Children would not be sent out very frequently (since this would be noticed by web admin) — So much time would be spent designing each child to send out on t. net.
Ray specifically avoided using Machine Code in "Ancestors", to avoid .07ff ... by using code involving an "interpreter Engine" that no other computer would have.

So Ideas I'm working on now:

15 1) I want to write summary of my recent work on implementing v "BZZ" to do GP: Phase I of TM

2) I want to write more on future of AI for Talk.

18. 3) T. idea of what kind of social activity to have in a A Life community designed to
19 Solve external probs/TSQ's. How to implement a prob-solving "culture" (147.28ff) → 168.00

20 4) Perhaps how to get this done on Web re "benevolent worms" or just volunteer time on "unused Machine Cycles".
1.8 in ... = x $e^{1.8} = x$
.1.2/yr .10 - .05/mo

Re: 3) Perhaps give more detail look at T Ray's work (described by Steven Levy's "A Life")

24 Perhaps look at Ray's writings (see his website in my "Bookmarks" on NP500). 5/24/04

25 147.30: Another direction of t. community: Ming Li (et al) has a pgm "PatternHunter II.PS C:\PS\" PATTERN HUNTER
This detects, counts Homologies betw. two DNA or prot seqs. These are "Similar sequences, between them" measured by match, mismatch & gap scores." They say they can do t. searches very fast.
Q is: just what do they look for & to what extent would this be useful in inducing patterns in t. pgms that denote "Useful functions"? T. Q's are → to those for t. utility
30 (for induction) of "Regular Expressions" ← (which are, I think, related to finite state langs)
↳ This pgm may be able to detect similar sub seqs in same order in diff't cands.

4TM

ALIFE 147.25 – 148.40

Space

00: 147.40 : A cand. has to be able to solve a problem himself, before he can present it to another cand. We arrange the problem rules" so that this is a useful challenge. e.g. One could just mult 3 (x − ai)'s together (using random ai's) & ask for a soln. to the resultant polynomial. — Somehow, the Challenger & challengee have to be on same level.

Cands can get "money" for solving toplevel probs in the "probem pool" of 147.30.

They can also get money from one another by winning "problem challenges".

A cand. has to make his problem look easy, but be hard. A challengee has only a certain amt. of time to solve a problem.

08 There is also big Q of how do we generate children? Maybe we just look at the total amt of money earned per unit time for each known cand., & we generate [illegible] Lsomehow, cands based on this corpus (like BZZ), with weights.

0 — We discard cands from population that are not in top "↑ percentile" of earning rate.

We may use ordinary GP method of crossmut for .08 – .10 — they may be better.

Another Q is just how much money to give for each soln — so the amount won is in a useful sense, a measure of how good that cand. is in problem solving.

Suppose cands challenge one another. If they know the "worth" (= rate of wealth growth) of all cands; they will want to pick a poor cand, since a poor cand is easier to beat.

17 The poor cand will not want to take the challenge, since he will probly lose. → (149.00)

[SN] Computer "Worms"! Use of Computers on the Net" to reproduce worms: This could be a great Breakthru in "ALife": The worms would be rather benign to hosts & use very little computer

0 capacity of host. To be resistant against Antivirus S.W. they would have to have varying structure — difficult to differentiate from other, common S.W. in the Computers.

But the viability is not the most interesting part. How to get them to evolve to be more useful

→ in useful ways, is main problem. The Evolving so as to be covert in order to detect/eliminate

→ would be part of the all over evolution. First list various ways worm could be more "invisible"/diff. to get rid of/undesirable to get rid of. : It may be a good detector/deleter of other "hostile" viruses. — May look for ways to ↑ usability of Computer for user.

→ Hvr Hackers may modify worm so it only does "useful" things & has no agenda of its own — could pressures of that sort "survive"? in the environment, might they find themselves? (probably would!)

— So, since both involve evolution, it may be very difficult to separate them!

Also [NB] early (est?) paper on Viruses showed that to distinguish between a properly constructed Virus & a legit computer pgm. is an "undecidable" task.

30 A "Friendly" life form could meet an "unfriendly" life form in a computer! If they fought One would kill the other. They might cooperate, or agree to both leave host because fighting is expensive. Possibly coexistence hvr, means sharing resources of host (which they don't want to do) (About to 149.00!)

REV: .02 on time to do 2-3-Tree w. PBCC / PB35 / Pent500 / Pent2800.

.01 : So, I should be able to make a 2-3 Tree pgm that works in PB35 that is very close to what is needed in PBCC; so I don't have to make many changes into the new environment.

.02 General Conclusions about speeds of PB35, PBCC, Pent3 500, Pent4 2800:
PBCC is about 1.3 times as fast as PB35 (145.10; 145.05 v.s. 145.10)
on PBCC, Pent 2800 is about $\frac{2.6}{1.4} = 1.85$ (1.857) (145.07, 145.10)
faster than P500 — Not much difference: Perhaps check this more carefully. My impressions were like 2.5 to 10. I expect P2800 to be 2.5 factor on 32 Bit PBCC.

If we use Virtual Memory in PB35, The speed goes down by factor of about 60.

We can use large amts of RAM in PBCC w.o. slowdowns however.

---

0 For time needed for various corpus sizes (& lower Bnds on times) see 144.18–.28

T. max corpus size for PB35 to use normal Memory is 7.85k

Using Virtual Memory 2.85 M (but speed ÷60)

Using Pent500 in PB35 & small corpus, it takes 10 us to insert 1 symbol for corpus length 10.

.14 Time/symbol ∝ ln 10; so corpus length of 10^n takes 10 n usec per insertion.

A "regular expression" is a form of a NGMST ≡ d language.

SN In Unix/Linux There is a pgm that searches for texts satisfying a "Regular Expression". Recent Content is one kind of "regular expression" That can be searched for. However many other types of expression are searchable: We should make a big project of finding kinds of regular expressions that are (a) findable rapidly (b) useful for prediction. See folder C:\PS\Tries: New start of paper on "Regular Expressions" — Ray see used (3/15/04) external paper as well. → 149.25

(reg. expressions may be a kind of FSLang. (≡ FSG Grammar.)

0 **SN** My impression has been that ANN is good for predictions involving non-linear continuous functions. This Guy in Bio informatics says best pgms for predicting structure from DNA sequences, has been a ANN pgm. — which seems to be a discontinuous problem! — But I would have to know much more about what the pgm does!

---

28 **SN** ALIFE: "Avida" at CalTech, (see website / Bookmark) They have This Pgm (freely available)
↳ Each individual has its own pgm & time share: Individuals are not "open" to other individuals. — There are some of the devices pgms are "Hidden". Each has own section of RAM They don't share Them.

To create organisms (pgms) from small no of insts. I dea is to use ALIFE to solve an external problem:

30 — A way I would do it: There is a pool of probs of diffing diffy: Each has a labeled "Reward" if solved. We somehow arrange the making of Mutations & a reproducing algm so that There is a pressure to "Get max reward": which means "solve hardest probs". We may allow creatures to eat one another, to get Parts to make "strong" (= reward getting) children.

It would seem (tho there could be no point of "Cannibalism", since one normally takes the genotypes of successful cands & mixes Them w. genotypes of other successful cands. Cannibalism wouldn't yield anything better!

T. possl. advantage of an ALIFE problem Solver: T. cands can present problems (of comparison) to one another. This is an automatically updated TSQ. A problem is to get the problems involved in surviving in the society, to be related to the top problems presented to the system.

Perhaps the social intercourse consists of cands presenting problems to one another!

5.27.04

4 TM

2^{32} $2^{30} = 10^9$ so 4G = 2^{32}.

running BigMem.bas in PBCC.

Dim A(268000000) As Byte. 268M

T1 = Timer

For x = 1 to 250 000 000 250M

;etc.

It took 105.131 sec. ! ~ 100 times slower than BigMem4.Bas. } still very strange!

T. Diff. betw. BigMem4 & BigMem is rather small!!

OH! I did 250M loops rather than 25M — so 10 times as long — not 100 times.

Try BigMem w. 25,000,000 only ; Also note: Disc does a lot.

10.1065 sec!

Try BigMem4: 2.64" only! The thing that slowed down BigMem.B: (.00)

I use 268M in A() ; BigMem4 uses only 26.2M in A(0)

try 200M in A. for BigMem, only 25M loops —

Took 2.42 sec ; so using > available RAM, it used the disc a lot.

So remember: Don't use so much RAM! 200M is probably fine.

I may get 512 Ram, hvr.

.17: 145.40 on PBCC: "DATA" and "READ" May be somewhat differt from PB35

→ No such word as "SHARED" listed. ← This may be serious Diffy.

∃ a "VARPTR" function (but not Varptr 32 — I guess its o.k. — it returns a 32 bit address. — see keyword VARPTR

I looked up "Function" "E Ti word "Export" makes a function accessible to other pgms. (May be irrelevant)

See "Global" vs. Local. → read this section carefully before using "Global".

That's it !

READ$ → "Read$" : only ascii is readable. See if there is a way to get numerical data in. !

Other way: I can write in contents of t. first Node. Or have a file containing t. data, that it uploads.

For setting up t. initial contents of t. first Node, use a seq. of assignment statements:

put it in a srtn so it can be reset when necy. In PBCC, I could just use MAT assignment of MAT A() = B() ; where B() is t. array w. t. proper initial Node Values.

T. System may have a Read stmt! try it out!

Read$(7) is a (string) of t. 7th Data object:

This can be converted to a number viz Instrs Like CVBYT, CVD, CVWRD, etc

So no real problem.

Also STR$ and VAL (& MKI$)

Actually | Val converts string express. to no. | This is what I need to put numbers in DATA statements.

I think to try to VAL can have decimal pt. like "38.753." (or even commas, I'm not sure about commas — leave them out.)

— see "VAL" keyword. like "33,923.13"

PB35 has Val & many of t. assoc. functs! So I can do t. changes within PB35 & test them, before move to PBCC. Hvr, PB35 does not have "Read", it only has "Read$"

5.27.06
4PM

.00 : Speeds of various Basic pgms with Pent3, 500 vs. Pent4, 2800!

1). To do loop of A(i,j) = A(i)+1 for 25k times.
The do it 25M times 4 times each so 25M operations,
.04 In PB35 this took 3.4 sec ± .05", on HP500
.05 so $\frac{3.4}{25}$ us = 136 ns/op or 68 clocks per op.

.06 2) To do 25M/ operations same in one loop using 26.8M byarray
.07 In PBCC took about 1.4" ± 2" on Pent2800
So 56ns per op. or 56 × 2.8 = 156.8 clocks/op

$\frac{\text{Pent 2800 speed}}{\text{Pent 500 speed}} = \frac{2.6}{1.4} = 1.857$

.10 3) to do each of .06 on ≡PBCC HP500 took 2.6 sec ± .25sec.
or 65 ns per op.

so PBCC is ~ $\frac{3.4}{2.6}$ = 1.3 times as fast as PB35

So it looks like PBCC can do large memory w.o. slowing down — so we could get factor of. $\frac{3.4}{1.4}$ = 2.4 so using Pent28 & PBCC will increase speed & structures

So Conclusions of 144.18 - .28: speedup by 60×2.4 = 144 over 144.18-.28
.16 1M symbol corpus would take 60min → $\frac{60}{2.4}$ = 25 sec. for 1Msymbol corpus.

Maybe not so BAD! See 145.18:40 → 146.17ff

.17 Unfortly, the porting of a pgm from PB35 to PBCC is extremely non-trivial! The syntax is
.18 often slightly different and diff to find! The printout of error(s) into compiler is interesting:
.19 it can give a number like (10,7) which means 7th word in 10th line.
.20 There is something in PB35 about reference to Arrays via SRTNS being by "value" rather than "?". I don't know what this means — Also the 2-3T pgm does use a srtn.
a/or defined functo, & it does work ok.

~~First Transport SN 14 (Re pum 2-3T pgm)~~
~~Re " SN Re~~

Just try transporting the pure 2-3T pgm. Make a contents of what the various SN pgms are:
On top of SN18ev.Bas Rooms a doen of same old pgms

>SN14 may not have reinitializetion of the arrays to zero — so it works once, but can't be repeated. (w.o. zeroing). ∑ It may be easy to zero an array using the "MAT" command in PBcc.
The most obvious difference PBCC is first line is FUNCTION PBMAIN()
last " " END function.

SN14 M 32 bit Dos,
Sort < Myfile.txt >
Sorted.txt.

.30 In the "Documents online," the first line is always "Function PBMain() AS LONG"
My experience is that "AS Long" is uneasy, Try deleting other parts of the line..
See: keyword "WINMAIN function" — for alternatives to "AS LONG".
If there is any printout, I think its necy to add the line!
10 If a INKEY$ = "" Then goto 10 ; otherwise the screen will disappear as soon as the pgm ends.

The technique, is to write the pgm, change it till it compiles ok. (no "run" needed) (Compilation might may have useful feedback as of .18 - .19) After compilation, it is possible to then the display & go to the "CCDir"; do a refresh & double click on the "exe" file — usually second from the top.

→ 146.17 spec

5.26.04

Speeds of My 2-3 Tree ppm

00 The ×60 factor seems familiar to me! ~~...~~ Just what I can do w. That factor, is unclear!

Now in virtual mode see how big H1 can be.

try ~~#~~ keyMx = 5 Hy = 10 k. k... = 5 gets −30 as output! | 6 → 346
next +527 7 → 1476 | 6 → 350
7 → 1463 | 7 → 1718

Hy = 100k 7 → 1698 | Hy = 1M 7 → 1703 | Hy = 10M 7 → exp. many errors.

Hy = ~~5~~M many errors ~~Hy~~ (Hy = 2M ok) 7 → 1465 Hy = 3M no good Hy = 2.5M ok. 7 → 1700

Hy = 2.7M ok. Hy = 2.8M ok ← BUG! see (.29) Hy = 2.9M error Hy = 2.85M

09 So Hy = ~~...~~ 2.85 ok (1.58M is more likely → see .32) Hy = 2.9 bad Pretty close cut. 20 × 2.85 = 57 Megabytes!

10 This is >> the 16.777 Mby Peat Fre(−1) said I had!

For Keymy = 10 I got 126256. (≈ 121106 of 143.40R) so it may be working.

~~AM~~ It was ~ 10 μs/symbol ~~in normal~~ using normal (ram! so .6 ms/symbol in virtual memory.

~~Found~~ so .6″ for k symbols 600″ = 10′ for M ~~symbol~~ symbols 6 minutes for 600k symbols.

2.8 minutes for 2.8M symbols — so I can use this to compare w. 1M corpus.

or 600k corpus.

So I can do ~~...~~/2d stuff on Colgary Corpus

18 Conclusions! In normal (PB35 memory 7.85k corpus is ~ Max size (143.40)

Using Virtual Memory, 2.85 ~~M~~M corpus is possible, but speed is ~ $\frac{1}{60}$ of normal memory.

20 Normally it takes 10 μs per symbol for a corpus of length 10, and time ~ N ln N.

So N = 100 is 20 μs/symbol. | N = 100k is 50 μs/symbol
1000 " 30 μs | 1M " 60 μs/symbol.

So a 1M corpus would take ~~.06 ms/symbol ... 60 ...~~ 36 μs/symbol → 60 sec = 1 Min.) → 25 sec Using P2.8 and PBCC

25 ~~...~~ . Using virtual memory it would take 1 hr.

Using ~~#~~ Pent4 2.8G (2.5 times as fast) $\frac{60}{2.5}$ = 24 minutes for 1M symbol corpus. → 500 145,00 −.16

.6 × 24 = 14.4 minuts for 600k corpus.

28 These figures will be mult by maybe 2 for rest of calculation.

.29 (.30) BUG! Error in computation of Memory needed for 2-3-T for large Corpi:

30 — I. present ppm. Uses words for addresses. This is 16 bits 2^{16} = 64k only. For corpora of > 64k.

We need 3 or 4 bytes/address. This will ≈ halve the size of compressable corpus:

I had 4 byte ~~and~~ arrays, ~~#~~ 8 word arrays so 4 + 16 = 20; now I need 4 + 32 = 36 bytes/character.

.32 57 Mby (.09) ÷ 36 = 1.58 M characters in max corpus

Use of Double words may slow down PB35 further!

Perhaps try using PBCC (console compiler — it runs at 32 bits!)

time some Pgms in PB35 v.s. PBCC using P2.8. See if there is much difference.

In PBCC 250 M by available so corpus of $\frac{250}{36}$ ~ 7 M symbols.

TSQ .04,.12

250142.28

150K 500 kps
300/kb
10/hr:
50hrs.
~2days.

30 : : **NB** I was thinking of using **BZip2** output for estimation of 8 bit cost; but result is only ±4 bits: factor of 16, which isn't good enough! Looks like I will have to get (cost) pgm. Quicksort or Heapsort. / As is, I understand how 2-3 Tree, Qsort work. I want to understand Heapsort also. It may involve Tree structures.

of! 140.40 **142.28** ⇒ .40 is a nice "Hy level" approach to TSQ design construction: GOOD TO KEEP IN MIND!

35 Try to see how various func. types, prob types, soln. types can fit into this scheme. → (.12)

SN Under what conds can we use ALP & get preds of known expected error yet not use "Cross Validation"? Perhaps if model construction is entirely a priori: we hadn't seen data when pgm. was written: Exactly what does this Mean? After pgm is run on new (previously unseen) data, we have a certain bcost

0 for bit of original data. This is expected value of future; (next symbol); bc/symbol. → (151.12)

If seq. is assumed to be "stationary" Then assignment of bc/symbol to a subsequence gives a ≈ "unbiased" estimate of bc/symbol.

12: (.05) → One way is to write tsq "in English" at Hy level. Then put in problems and solns. — of TSQ proper "(i.e. not (yet) pre corpus)" Both in actual code. Both probs & solns. should be expressed in "factored" form as much as possl., using sutns, concs, defns, that I think are "Generally Useful". Then each of these concepts ("factors") is either to be lrnd by TM via a tsq. or inserted into t. lang. This insertion can take t. form of a large no. of (prob/soln) pairs that use these concepts, being made part of t. pre corpus, that pgm codes before TM starts lrng. This "pre corpus" gives PPM lots of important "contexts" & their likely Contns.

A defn of a soln. could include "Use hour #18". TM will have lrnd to correlate t. various

20 hours w. various problem defns. Hour #18 would be followed by various choices used in t. implementn. of t. hour. For TM to lrn to devr hours, give it tsq's of a list of probs + solns, that could have used a particular hour — so it could both invent t. hour, as well as "rules" for its implementation.

① The initial TSQ's are to Teach TM elementary skills: Mainly lrng techniques.

② Mainly to teach Me how to design TSQ's to get TM to acquire skills of various kinds.

③ Ultimately to get skills needed for Phase 2

→ I could take t. present 2-3 Tree pgm, see how long a corpus I can do in

30 poor Basic Ram; then see how slow it gets for corpus using PB35 w. larger Ram — My impressn is that its like using a HDD for t. data, using a RAMdisc — perhaps acceptable for many "projects".

See ~81.00 ff for studies of speed of that 2-3 T pgm. T. pgm. being used was SN18B.Bas & also SN18C.Bas : I think t. arrays use 20 bytes per symbol of corpus. PB35 allows maybe 200k for arrays! so corpus = 10k symbols.

SN18B.Bas H1=1000; DIM DN(H1) as word. (← 16 Bits) So: Dynamic Memory?

H1 = 10k	(out of Memory)	6k	ok	7.9k	out of mem
2k	ok	7k	ok	7.85k	out of memory
4k	ok	7.5k	ok	so 7.8k is ok.	
8k	out of memory	7.8k	ok	7.85k is out of memory	

So I may be able to do a 7.85k corpus

In SN18CV.Bas All arrays are virtual. I want to know time. Maybe 100 sec or more! For KYMX=10 try KYMX=5 output = 15.958 I think it doesn't work for KYMX much < 10. For KYMX=10 I got 121106... / 2115 w.o. virtual vs. ...

1.19.04
1.29.04

IMPT → TSQ's & Reference UMC's .28 ff to .40 at least ~~2~~ 143.04, .12

PPM (objection / Criticism) (.02)

.0 : p 235 of Num Rec: Says for Quicksort, worst case is input array that's been already sorted! I don't see this!

.02 (SN) Possl. Criticism of PPM! That "con" contexts that only occur only 2 or 3 times in a very long corpus, are unlikely to be greatest pred. ~~predictive~~ — (From the test work on A Z(?)) — see letter to Wolff
If a second layer of definitions is used, t. freq. of symbols should be based on t. "most likely" parse — i.e. not all occurrences of ~~a~~ ab (when α has been defined) will be instances of α. A certain no. of them (e.g. r_b of them) will be due to "coincidence".

On Q sort: 1) we can (perhaps) signify the speed of comparisons by starting comparisons of 2 strings on the r^th symbol. r=0 for t. top & bottom "chunks of text"; for other chunks one always knows 1 string that is (≤) all strings in t. chunk, so compare these 2 strings to get r. 2) Horowitz & S. get t. expected value of cc for Quicksort soln by solving a difference eq. Prob'ly one can get difference eq. for σ^2 & get vars. of d.f. — which would be very useful!
At first Glance, t. difference eqns for expected values & for vars. seem identical: so σ^2 = expected value! But check this! My impression was that t. D.F.'s looked like ⊔ rather than ⌒

(No!) Corresponding to 141.12: $\overset{\sigma\equiv\sigma^2}{S}(N) = \frac{2}{N-1}\sum_{i=1}^{N-1} S(i)$. — Note absence of "(N)" term (Horowitz may have gotten "N+1" for that term, but it's irrelevant here.)

$(N-1)S_N = 2\sum_{i=1}^{N-1} S(i)$, $\quad N S_{N+1} = 2\sum_{i=1}^{N} S(i)$

$N S_{N+1} - (N-1)S_N = 2S_N = N S_{N+1} - N S_N + S_N = 2S_N$ ∴ $N S_{N+1} = (N+1)S_N$

$N S_{N+1} - N S_N + S_N - 2S_N = 0 \quad N S_{N+1} = N S_N + S_N = (N+1)S_N$

$\frac{S_{n+1}}{n+1} = \frac{S_n}{n}$ (unless n=0) so $\boxed{S_n = kn}$ — $\frac{S_N}{N}$ is this only soln? Yes. → $\frac{S_x}{x}$ = constant ∴ $S_x = \ldots$

So σ^2 grows less faster than E(n), so $\frac{\sigma^2}{E}(n) \to \frac{k}{\ln n}$ $\quad \frac{\sigma}{E} \propto \frac{1}{\sqrt{n}\cdot\ln n} = \frac{\sigma}{E} \propto \frac{\sqrt{N}}{N \ln}$

.28 TSQ's: A Major problem is to design (inst set // ref. language) so that t. heurs I want TM to use can be easily expressed in that Language and/Or t. lang is "extensible" so th. needed concepts are easily definable. In any ~~adequate~~ universal lang, I must be able to define
.31 (or equivalent) concepts so that .28 is satisfied. It amounts to putting certain impt. kinds of
.32 a priori info into t. Ref. Machine → (e.g. see (.28))
.28 ff may be t. main principal of TSQ, Ref machine design. T. rest of t. "Learnability" of t. TSQ should be implemented via (Bern seq) / (PPM) type [illegible].
So I write the Seqns of probs (& possible Solns), & get see what pc's PPM gets for my solns. [I don't need to program it for this — BZip2 itself would be adequate!]
.38 One way to implement (.31—.32) would be to give TM certain set of (prob/solns) "free" as part of its ("pre corpus" / a pri corpus) → (See 143.04)

1.19.04

4TM

: One Q is whether I can do w.o./ 2-3 trees pgm: i.e. just use the Heapsort like BZIP2. See if I can find it.

Quicksort is not bad! See bwt.cpp in c:/ps/BW/xml – Nelson code

Also look at assoc HTML doc & z's bwt.cpp

Seems easy to pgm: See Horowitz, Sahni p 121 for quicksort — also, NumRec. p 233

I don't yet see why the speed is needed! → It is usual int. simple, recursive form of Quicksort

.07 Re: Quicksort: If the partition pt. is chosen at random each time, & $E(N)$ is t. expected time to

.08 — Sort a list of size N, then $E(N) = \frac{1}{N-1}\sum_{i=1}^{N-1}\left(E(i) + E(N-i)\right) = \frac{2}{N-1}\sum_{i=1}^{N-1} E(i)$.

One soln is $E(N) = 2N$: $\sum_{i=1}^{N-1} 2i = \frac{N(N-1)}{2}$ so $\mathcal{E} = \frac{2}{2}\, 2N$. $1+2 = \frac{3(2)}{2}$

which seems unreasonable!

Eq. .08 is wrong. After a random point is chosen, we have to do ~ N comparisons, which takes

time: then add in .08 : Assume exchange time << comparison time so

Hands p12 & (eq. 3.4) get slightly different eq. $E(N) = N+1 + \frac{2}{n}\sum_{i}^{N-1}$ (perhaps)

12 $E(N) = N + \frac{2}{N-1}\sum_{i=1}^{N-1} E(i)$

Guess $E(N) = a N \ln N$.

(N) Time of comparisons here depends on ordering of data. Those applies to Horowitz's proof!

$\sum N \ln N$: consider $\int x \ln x$ $\frac{d}{dx}\, x^2 \ln x = 2x\ln x + x$ $\int 2x\ln x = x^2 \ln x - \frac{x^2}{2}$

$\int x\ln x = \frac{x^2}{2}\ln x - \frac{x^2}{4} = x^2\left(\frac{\ln x}{2} - 1\right)$

so $E(N) \sim N + \frac{2}{N-1} N^2\left(\frac{\ln N}{2} - 1\right) = \frac{2N^2}{N-1}\frac{\ln N}{2} - \frac{2N^2}{N-1} + N$

$N + \frac{2}{N-1}(N-1)^2\left(\frac{\ln(N-1)}{2} - 1\right) = N + 2(N-1)\left(\frac{\ln(N-1)}{2} - 1\right) = N + (N-1)\ln(N-1) - 2N + 2$

$= (N-1)\ln(N-1) - N + 2$.

So what is $\sum_{i} i \ln i$? $\frac{d}{dx}\, \frac{x^2}{2}\ln x - x^2 =$

$\frac{x^2 \ln x}{2} - \frac{(x-1)\ln(x-1)}{2} - x^2 + (x-1)^2$

$-2x + 1$

1 2 3
N

1 4 8 $12\frac{1}{2}$ 18.833   X, Σ
1 5 12 $24\frac{1}{2}$ 43.16
$\frac{13}{3}.2 + 4$

$X_N = N + \frac{1}{N-1}\sum \text{old}$ $X_{N+1} = N+1 + \frac{1}{N}\sum^{N} N_i$

$\sum_{\text{new}} \to \sum_{\text{old}} + X_{\text{new}}$

$X_{N+1} = N+1 + \frac{2}{N}\Sigma_N$ $\Sigma_1 = 1$

$\Sigma_{N+1} = \Sigma_N + X_{N+1}$

S=1
For N = 1 to 100 (step)
X = N+1 + 2S/N
print N+1, X, X/(N+1)/log(N+1)

For N = 1 to 20: Σ = 1
X = N+1 + $\frac{2}{N}$Σ : print N+1, X.
Σ = Σ + X

$N+1$	X_{N+1}	$X/(N+1)/\ln(N+1)$
2	4	2.88
3	8	4.6
4	12.6	5.23
5	17.3	5.6
6	23.4	
7	29.3	2.15
100	937.47	2.035
200		2.03609
500		2.025
1000		2.0225
10000		2.01677

So for large N $X_N \approx 2N\ln N$.

This is "Expected value". No N^2 is given! If t. "pivot pt." is chosen at random, it would seem that t. initial ordering of t. data would not affect t. result!

100 : 350 100 30
200 360 10 50 30
500 250 110 20 10
1000 225 27.5 10 3
10000 16.7 57 1

$250 - 20x$ x = 2.5
$225 - 10x$
$167.7 - x$
$165.2 - 0$
$2.01652 + \frac{1}{N}$

N.B. For kinds of Difference Eqns can be used to get σ^2 at each p.o. of cc of soln.

00 : (SN): For a T.S. of 0's & 1's only: How close is a curve that mmz MS error to using that curve & pc & Maximizing pc?

I.e. Say $S(t) = 0,1$ is the time series:

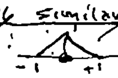
03 1) to find $f(t) \ni \sum_t (S(t) - f(t))^2$ is min. This = mmz $\sum_t -2(S(t) \cdot f(t)) + f(t)^2$.

max = $\sum_t 2 S(t) f(t) - f(t)^2$ $= \left[\sum_{t:\, S(t)=1} 2 f(t)\right] - \sum_{\text{all } t} f(t)^2$.

06 2) to find $P(t) \ni \prod_t P(t)^{S(t)} \cdot (1-P(t))^{1-S(t)}$ is max.

07 $= \prod_{S(t)=1} P(t) \cdot \prod_{S(t)=0} (1-P(t))$

→ Hvr, (.03)? can also be viewed as a max pc model w.r.t Gaussian D.F. — So check this out!!

HA! So They are both very similar. .03 uses the Gaussian D.F. for error. .06 uses a linear d.f.  v.s. 

(Maybe ln of this ≈ as opposed to mean sq. error. : we want penalty = ∞ If we say "1" when "0" is correct. → $\ln(1-|x|)$ x = error.)

→ There is some Hanky Panky w.r.t Gaussian d.f. since we do not actually state σ^2.

One "Lazy" way to solve .06 is to solve .03 (as first approx.) then recent approx. moves the data pts in accordance to .06 using Gaussian error & ... simpler soln. We again move the data pts. & use the .03 soln. and so on. This converges to correct soln to .06.

(Maybe $\ln(X(1-x))$ is more exact, (and .07)! The (.06) is exact.)

SN Instead of PPM, I could use that old ≈ 2(4) method, in which I only make 1 definition, using the recent context. I do this for all of the contexts that have occurred before (ssz > 2) & use a wtd pred. of all of them. While this may take Much longer, it may be more accurate.

It may be able to help get regular PPM more accurate, by telling how to wt. preds of various context strings.

I do have exact formulas for pc's using factorials & summations. Hvr, my algebra approximations may not be correct. Check using Maple.

I would like to compare it to PPM.

Actually, it doesn't really implement "definitions" (to create new symbols from old) any more than PPM does. I am hoping that PPM will be good enuf to get me thru Phase I. Another way to get thru phase I is by GP.... possibly using PPM for Lsrch — which is a bit like Using a "second order" Phase 1 — The induction method used is Universal, rather than the limited modeling of PPM. In such a system, the lowest level TSQ must have problems

.10 that are like extrapolating a seq. of functions. I.e. in the QA TM, we have a sequence of Points that are sets of $Q_i A_i]_{i=1..n}$. A funct. for each value of n: = $f_n(\cdot)$ — we want the TM to (rapidly) extrapolate a PD on Fns. PPM is able to do this rapidly, but much better performance is prob'ly poss'l. Actually Phase 2 sort of does this.

(We treat this as an "Unordered Seq" .... Tho it is ordered. As "unordered seq": it is a set of approxs to a funct that will work for all "All QA's".)

We assume that Phase 1 has to solve problems that are Phase 2 meta problems.

Phase 1 could just as well apply such methods to its own meta problems of various kinds.

Hvr, at the beginning, Phase 1 will simply "solve problems" — to train Me as well as TM.

· I will learn how to write TS Q's & TM will learn to these TSQ's.

5.15.04
4TM

REV

Mon, Thurs, royal East
Soft Shell Crabs.

At least 2 impt items
1) Implementation of PPM & using it for Lsrch.
2) TSQ writing & how affected by (1).

201.138.40 O.k. : Some things to include in t. "outline"!

1) Explain. of why we decided initially to do 2-3 Tree ppm!
   a) N log N time.
   b) It could do insertion
   c) " " " deletion.
   d) It may be available as a done ppm.

Apparent faults of 2 3 Trees:
1) My ppm uses at least 20 Bytes per symbol of corpus, & 36 bytes per symbol for a large corpus. (144.29) — for a small corpus

1.5) Discuss recent work on how 2-3 trees may be unnecessary: 138.13

06 2) Perhaps initial discussion of "PPM": why it looks good! — Initially Idea was in Costa Rica. ~ Nov 2003; 3TM 421.13 Mentions BZZ as possy: so < But 3TM: 405.31 start. 405.12 ... 3TM 411.30 ... Bibl. review: 150.02

Q: just what were t. nature of t. "improvements", in terms of t. general problem of sequencial predn. of $\vec{L}$. — problems of computing kernel params & expressing kernel { T. Improvements to PPM are usually the "Tree" language — which is really comfy

2.5) The kernel approach: how to update t. kernel cheaply! How to get individual token PC's cheaply, comparable w/M.!

3) Discussion of 3 kinds of Lsrch TSQ, ||, (random MCarlo)

0 Why || & MC look much better at first (factor of k or 2k where k is length of soln in tokens)

Conds under which this factor does not occur.

So discuss conds under which TSQ seems best.

4) The soln of t. "frequency of updating" problem! and T. possl. use of Heapsort (?) or quicksort rather than 2-3 Trees. (Initially, I may want to use 2-3 Trees)

Go thru recent work & see if t. rest do indeed cover impt. ideas.
→ 142.28–.40; 143.04, .12. || 14.30 15.20 on TSQ Report

Also Go thru recent recallings of ideas on TSQ's — Partly modified by recent ideas about PPM as a cheap [redacted]. low level predictor.

Discussions of adequacy of PPM in this applicn. 138.04, 132.20 & in particular (PPM does not have to be Universal!)

0 Q: Re: Improvements in predicting E & $\vec{L}$ sequences, Can all possible improvements in t. guiding PD be affected by ?? Perhaps/almost because $\vec{L}$ has just about all info about original corpus: But not all.
It takes very little extra info to go from $\vec{L}$ to t. corpus

3 → Could we apply PPM to $\vec{L}$ w. any signift. compression? It would be easy to apply PPM to t. $\vec{L}$ sequence!

→ It would be really easy if I could get the BZZ code & find t. part that generates $\vec{L}$.
I have no idea as to whether this would work!

Maybe read original BW paper: it had lots on mechanics of sorting.

It may be that .23 is sensitive to just what variety (improvement) of PPM one is using. For some, t. recursion may do nothing — for others, it may do a lot, others it may have negative effect.

No! on second thot, all varieties of "PPM" do t. same in that they convert corpus to

0 $\vec{L}$. T. differences are in how they try to predict $\vec{L}$ — so for several recursions of .23, we only end up using t. "different version part" of PPM once.

We can try various methods of predn. on $\vec{L}$: linear & non-linear (as mentioned before).
33 Even try Lempel-Ziv.!

Actually, I'm not terribly optimistic about recursing PPM, but its so easy to try, that it certainly should be tried! So when I have t. $\vec{L}$ generating ppm, I will want to try these tricks. (.23 — .32!) → However, see (154.13)!

... has tried to respond ≈ "like Maple", doesn't know enuf for us to try to get it to try to understand ... about what it has lrnd? T. Maple-like machine can be given problems of a difty "solve", "integrate", "simplify", "prove", etc

(SN) Back to an earlier Q on t. "adequacy" of PPM for Lsrch: As an extreme case, Say we used full Univl dist for Lsrch over successive problems in a TSQ, using solns of previous problems as corpus. This would recognize all possl regys in corpus & use them to speed up Lsrch. By using a less optimum P.D. to Guide Lsrch, we endup w. solns of smaller pc & ∴ larger srch times

T. main idea of $TM_1 = TM_2$ was that we could somehow make t. problem of improving t. Guiding P.D., ~~problem for the~~ a part of t. normal TSQ. The Phase 1 → Phase 2 is one way to try to implement this.

(SN) on t. need for 2-3 Tree & its large need for ~~[illegible]~~ RAM — "T distribution sort" = RadixSort. or "Inner Loops" chapter 15, p301 — This uses some RAM, but not near as much as 2-3 Tree.

If we put stuff in lex order cheaply, as in BZCP2, say, ~~[illegible]~~ could use the $\overleftarrow{L}$ & corpus for prediction of individual Tokens w.o. having to insert anything into $\overleftarrow{L}$. We would have to know ~~[illegible]~~ where to insert a particular context string, hvr. If t. suffixes were in actual lex ~~[illegible]~~ order (by address), we ~~[illegible]~~ can do a binary decision search taking ~ $\log N$ decisions.

Say we have 2 tables both N ~~[illegible]~~ entries long. ~~[illegible]~~ T_1 & T_2. Dm $T_1(N)$, $T_2(N)$:

$T_1(\ell)$ gives the Lexical order of t. ℓth shift of t. corpus.

$T_2(z)$ is inverse of $T_1()$; It gives t. shift no. of t. zth lexical context.

To find where a context Y fits (its order no. will usually (always?) be ≠ integer): We compare Y w. shift no. $T_2(\frac{N}{2})$. If its before, t. next comparison is w. $T_2(\frac{N}{2} - \frac{N}{4})$; if its after " " " $T_2(\frac{N}{2}+\frac{N}{4})$

So we find Y's position after ≈ $\log_2 N$ comparisons.

~~[illegible]~~ So we find ... we can generate table $T_2()$ in N steps.

Given Table $T_1(\phi)$ we can generate table $T_2()$ i.e. for $i = 1$ to N: $T_2(T_1(i)) = i$; Next i

This would be good for periodic Updating of $\overleftarrow{L}$. Like Eq of 124.33

The comparison process could be speeded Up somewhat by noting in comparisons how many common symbols occurred before there was a dis parity. This would give idea of whether one should jump, say to $A+\frac{N}{16}$ or, if it is much closer, to $A+\frac{N}{100}$ or if it were more distant, to $A+\frac{N}{10}$, etc. → 162.04

5.13.04
4TM

0 • Consider Lempel-Ziv: compress = $x\frac{1}{2}$; v.s. Bzip2: $x\frac{1}{3}$. For Asci; $x\frac{1}{2}$ is 4 bits/symbol. $\frac{1}{3}$ is 2.67 bits/symbol. ← Actually its more like 2.2 bits/symbol for Eng text & LZ may also be better so difference is maybe $(4 - 2\frac{2}{3}) \times \frac{2.2}{2.67}$ $\frac{4}{3} \times = 1.1$ bits/symbol. For a 10 symbol pgm! This is x 2^11 = 2048! So it does pay to use Bzip2 rather than LZ. The performance on Bzip2 may or may not be very signif, hvr. — I'll have to try this out.

• In the Calgary Corpus the compression factor for LISP code wasn't very good, but my "Lisp" "Lispcode" will be (I hope) a fairly redundant corpus!

It looks like I will have to use PPM for generating Cands anyway — Re C miss get so useful feed back from PPM (Bzip2) pgm.

So First: write/review of how I will use PPM to do Lsrch. — why I will use TSQ, Then write it in PBasic. See if it can accomodate large out of files to do usual — It is possl. to get PB to accept large files, but it slows things down tremendously! Hvr, I may be able to tolerate this: the only large files I'm interested in are English texts; If it takes a day or week to do a file … still OK.

• For testing SeqUs. I probably will have files of < 1k symbols,

• But to test PPM on GP, I may need files of >> 1k.

19 SN Why was $\sigma^2 \to \sigma^2 \frac{N+k}{N-k}$ not a legit soln. to the "How Many Gifts?" problem? → N.B. $\sigma^2 \to \sigma^2 \frac{N+k}{N-k}$ only for k=1 … easy to show for k>1, not sure how to do it

20 The "proof" was that the factor $\frac{N}{N-k}$ gave σ^2 if present coefs were correct. That the "additional" $\frac{N+k}{N}$ factor was because coefs found were not correct. — That this eqn was true for Non-linear stuff, too, if "N was large enough — so …

So system was "locally linear" (would large N do this? — or would small σ^2 be needed?)

Re: .29: This seems closely related to PPM: we have various "Match lengths" corresponding to k: we could use "best match" (= MDL) or wtd sum (Univl. D.F.) 3 approaches to "HMC": ① .29 ② Get stats from previous corpus ③ use methods like PPM (which may be = ②)

26 TSQ's: One simple way to start: write a bunch of sample problems, & write their soln. pgms. / Not necessarily written w. teaching in mind! This is the start of the TSQ. It gives an initial P.d. for Lsrch on subseq. problems. But may be more carefully designed (scaffolded). We can use PPM for this TSQ, as it continues. It would seem that this is similar to that MIT "Henry …?" guy's pgm that learns to pgm. "by example".

29 In My case, all previous correct pgms (both given by instructor or solved by TM) for solns are used as "examples". See how his pgm works. It may have useful ideas. He has edited a book of papers by various people on this.

Re: 136, .15 (on 129½). It is certainly not clear as to just how we want TM to respond to problems in 129½ … How does it know what is wanted?

35 A kind of … would be to learn to respond properly to questions asked of Maple &/or Mathematica. So an inter… item would be to compare the "probs of 129½" w. those on Maple. Why is 129½ so unclear … "well definedness" of Maple's problems. Some Maple items: 1) Evaluate a function either numerically or have literals give values to params. ② Simplify (… ys to specify what characteristics one wants in "simplify". ③ differentiate ④ … ⑤ Solve ⑥ prove ⑦ disprove, ⑧ Maximize (wrt param set) — usually continuous, but

digital = Bool
not digital = linear
digital = Bool (= not linear)
#10 book questions re: digital.
Daniel R
12:30 Thursday
Re Bro/Sis.
Monday
7AM …
Try Practical … P28 — Bobs pretty 16 Bit

H.L

00 , w. 2 params: one would desc. behavior near x=0; t. other could desc. behavior at x=∞.
We could try 3 decays (6 → 4 params) to see if works any better.
Another poss is 1 decay. (w.o. $\frac{df(0)}{dk}=0$) so only 1 adjustable param.
I may be able to solve for t. 2 params analytically & then use it [as] first approach. for t. 4 param case(.

Hvr. I really need a good review of recent Menkes' work on PPM Predn, etc. & how to use it in Lsrch.

List of ideas to think about (≈ STACK):

0 1) Diffrnce betw OOPS lang & Lisp-ish lang. in Lsrch; Is it a fundamental diffrnce? Idea of doing part of pgm & having part of output. In impt Q is whether strings in pgms occurring in Lisp are more likely to be useful in induction than things occurring in OOPS pgms. Perhaps try them out — to get ideas.

15 2) TSQ's: 129½ is very interesting! It is a list of Mathematical "facts" That I expect TM would be able to lrn. A "fact" for TM is defined by how it is usd to help solve problems. Hvr, I don't think I ever discovered a reasonable way for TM to lrn t. "facts" of 129½! For each "fact", I want to say just how it affects TM's (behavior / problem solving Routines).
|| → .15 ff may be one of t. BIG Bottlenecks in TM TSQ writing!

10 SAAB was largely Concerned w. [these] diffys of this sort. I that that t. QA form of problems would cover an enormous range of problem types — so writing TSQs would be much easier... — but still not Easy!

I still want to write up a detailed outline on how to use ≈ "PPM" for Lsrch — But I think .15 is t. BIG PROBLEM. Having t. S.W. pgm for PPM would seem to make it easier to write TSQ's — but only of t. kind where I didn't know in detail, how t. lrng.
27 Was to be done. ÷ But Note (.31) 20 $\frac{2.5}{2.}=1.25$ $(\frac{5}{4})^{20} \sim e^{5}$ | $(1.25)^{20}=867$
My impression is that it's best to write TSQ stuff very hi Level (English), then work on down to pgmng.
One Top down way is to start w. Big Problem soln & factor it down to t. primitives:
0 If certain parts are too hard to factor! lrn to them as t. primitives.
31 (.27) → These TSQ's can be evaluated for incremental pc of each problem soln. — for trouble shooting we say get pc of a particular symbol. → This can be done by using Bzip2 as a BIT Cost w.o. my pgm saying — Still bit cost is ±½ so pc factor of $(\sqrt{2})^{\pm 1}$.
(Probly / Maybe) good enuf for troubleshooting!
[SN] in TSQ runs, we may want to put multiple (several (diffrnt solns to same problem)) solns. into t. corpus, so that TM could lrn from them.
[SN] If we ↓ bc per symbol from 2.5 to 2 : a factor of 1.25 Then 20 symbols — pc → $(86.7)^{-1}$; 10 symbols → $(9.3)^{-1}$.

5.8.04
4TM

I think Ray is ~~[illegible]~~ ≈ 135 ~~[illegible]~~ faster

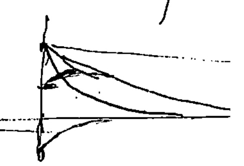
00 : 13430 : Also, writing these TSQ's will give me a better idea of what I want/need in a guiding P.D. (corresponding / analogous to PPM. One big difference betw. present state & 1990 (w Sarah) is that by using 4. QA TSQ, I can write TSQ's a lot easier. A TSQ having much lrng of Definitions, may make it easy to write them. In fact learning of definitions can replace "Telling" students things

The "Lap rule" Bern seq. seems to be close to the kind of suggestions for new Conds That I expect in lrng. There certainly are other impt. ways, but basically, I think that 4. Bern seq. is the main fundamental, elementary device by which (lrng. occurs / pd's are usefully obtained.) PPM is a rather fast way to implement something very much like "lap's" rule.

Lin 4000s
40
20 in HP500
17.5
7.1
4.3 — probs in fewer space
2.3
2.25
2.1

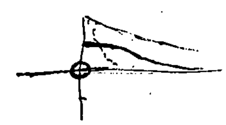
10 : 134.19 : The idea of these PD's on tokens as "data objects" or "Math'l objects" to be combined w. each other & w. various other data objects, to create new PD's. So we have an algebra of PD's to create new PD's from old (this is what much of Classical probability theory is about)! i.e. combining old PDs to get new ones. No good ways to get PD's from Raw Data.
In OOPS, 4. idea of associating a PD w. 4. freq. of use of Tokens in a particular problem soln. seems very interesting: — But mostly, 4. small SSZ, in a short loss of a statistical d.f., than a logical construct. i.e. 4. set of Tokens used in a PPM is an interesting, perhaps Useful, "data type" Learning to generate, manipulate these P.D's could be an impt type of Useful Metaproblem. Assoc w. it is learning how to generate & combine & manipulate S-funcs. — which it will need to know, anyway.

20By/by
2 G
100 Mby



20
2.1

(SN) A 2-param family of decay funcs w. nice properties:



$f(x) = a e^{-bx} + c e^{-dx}$. If we want $\left.\frac{df(\ )}{dx}\right|_{x=0}$ to be 0 we still have 3 params to adjust. For $f'(0)=0$ $-ab-cd=0$. so $ab=-cd$.
Another constraint: we may want $f(x)$ to = mean of $y(x)$ — so only 2 params.
We adjust 4. 3 params so that as a prediction $f(\ )$ has lowest pred pc for entire corpus. The score for each $f(x)$: is $\ln f(x)$ if $y(x)=1$; is $\ln(1-f(x))$ if $y(x)=0$!
so we want to maximize $\Sigma \ln(\ )$ or Max product $\prod_y f(x)^{y(x)} (1-f(x))^{1-y(x)}$: This may be easy to compute; tho we may want to use pseudo floating pt.

$f(x) \approx \frac{a}{b} + \frac{c}{d} = \alpha$: & $ab = -cd$ $a = -\frac{cd}{b}$

$ad + bc = \alpha bd$	$ab = -cd$

30

$-\frac{cb}{b^2} + \frac{c}{d} = \alpha$ | $c\left(\frac{d}{b^2} + \frac{1}{d}\right) = \frac{\alpha}{cd}$ | $\left(-\frac{1}{b^2} + \frac{1}{d^2}\right) = \frac{\alpha}{cd} = \frac{\alpha}{ab}$

No! (so $\frac{1}{b^2}$, $\frac{1}{d^2}$ & $\frac{1}{c}$ are linearly related so any pair of b, c, d will determine the third easily.) No!

For computation, we'd like to be able to pick values for 2 of 4. variables & then easily find 4. other 2. Say we know a & b ($= -cd$) $cd = -\beta$ $ad + bc = b \cdot d\alpha$
Say we know $a \cdot b = \beta$.

$(a-\alpha b)d + bc = 0$
$\boxed{(a-\alpha b) \cdot d + b c = 0}$ also $c\,d = -\beta$ so we know $\frac{c}{d}$ and $c \cdot d$ $c^2 = uv$
gives $\frac{c}{d} = u$ $c \cdot d = v = (-\beta)$. $\frac{c}{d} = u$, $c \cdot d = v$ $d^2 = \frac{v}{u}$

5.6.04

4 PM

→ Talk at Royal East.

4th June friday 

DO one way to compare Prams, compare pc of a soln: i.e. cost finding it via PPM v.s. via GP.

To compare Soln. times we have to consider long during Srch ..... which is what GP always does. This seems Diff't ! .... at a more "el." level, Compare t. "Goodness" of GP i PPM w. a fixed population. For GP, no lrng during a generation, i no lrng during a Generation for PPM as well.

One (perhaps) advantage of Lsrch (using PPM) [← The "using PPM" is not nec'y here.] is that short codes are automatically "selected for".

In normal GP, one has to modify t. "fitness funct" to include idea that short codes are better. Lsrch using same PD, does this automatically, — GP deselects (using fitness funct) a cand after trial because of excess length. Lsrch deselects before t. trial, which seems much more efficient.

One Big Q is PPM v.s. "OOPS". {Also Q of whether OOPS is really faster in Lsrch : (11 Lsrch) Does PPM really give a better pd for Lsrch?}

We have to consider various (i.e. A.H.) mechanisms that have been i can continue to be introduced into OOPS that could signify ↑ its power. T. extent to which I can do this in PPM is unclear. → (Also note 136.10)

T. Useful Mechanisms for induction in OOPS:

1) T. usual Lsrch's distribu. for t. next token (Reu Seq).
2) Ability to define new tokens
.18 3) Ability to (combine, use, modify) pd's on tokens used in previously
19 successful cands — to be used for present problem. → This seems like a v.g. idea → 136.10 → 135.10

It may be that PPM does do (3) (.18) to some extent — perhaps better than PPM ! it does this by suitable choice of "Escape" probys (or equivt). My "Kernel" does this ....

How effectively, is not clear.

(1) Is pretty done as well or much better by PPM

(2) In Costa Rica, I got idea that PPM would ≈ simulate effect of defns.

Now, here I'm not so sure! — Because definitions can enable coding names i various "production" languages.

Most Recently I had idea that it would be useful to try PPM (or my version of it) on English text (to see how it compared w. regular PPM & its refinements).

But more important: I wanted to try it on GP problems to see if it gave significant speedup.

→ So: What is expected Value of PPM ~ PPM for Lsrch? What do we expect to get from it?

1) It mite be good and of itself (2) to demonstrate useful TSQ writing; (3) to enable TSQ writing & testing, → 135.00

Well: Two results: Don't seem to fit together. (This is Re: Murry's Computer)

[Re: Murry's Computer E500]

1) I put Murry's original 4.3G HDD into my case ; replacing my 4.3 HDD or previously the 40 GB HDD.

On start up: I got to t. Windows Big Logo Screen i began a cycling routine (prob'ly on HDD) — Never got further. This is same behavior that was got using Murry's 4.3G HDD and Murry's E500. So it would seem that it's t. S.W. on Murry's 4.3G HDD.

[for Murry's computer 5/7/04]

2) I formatted t. 7.5 GB HDD i put into Murry's Hardware E500 ; Installed W98 F.E. w. no trouble, but on reboot : a total mess! (maybe I didn't prepare HDD properly?) Suggesting that E500 HW is at fault.

{Surprising!}

4TM

.00 : Note that in all of the forgoing discussion, the fact that PPM couldn't detect regys like 131.09–.13 is quite irrelevant.

.02 A discussion of a soln. to 132.15 ff is in the NIPS report: The section on the R (recognition functions) — In the first or second section of the report (page 8 of 4/11/04 document)

.05 So the point is, I really shouldn't worry (now) about 131.09–.13 — that the approaches of 132.27 ff & 133.02 are more to the point.

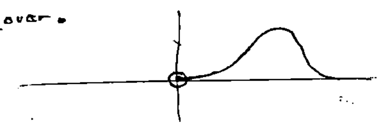
A more General Q: Certain Mutation/crossover algms have a greater expressive power of than PPM in some respects. How to characterize this: Poss. ways to ↑ Power of PPM.

.1 Consider 2 parents that are function trees. Consider all Koza-type crossovers. This gives a uniform d.f. over a finite set of kids. Considering all the G pairs of parents, we get a D.f. on the "next Generation"

How does this Df differ from those obtained by PPM?

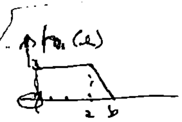
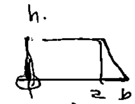
The GP distribn: w. a finite population, we only get a d.f. over a very finite offspring population — kids are restricted to being found of 2 parents in the population, crossed over.

In the PPM d.f. all possl. (finite) strings (≡ pgms) have proby > 0.



The GP D.f. is formed by the concat of 2 d.fs. Chose a parent at random, chose a crossover pt. at random. The top end of the crossover is one D.f. the bottom end of the crossover is the other d.f.

Perhaps (it looks like): if $f(\ell)$ is the distribu. of length of pgms, then after crossover the D.f. of length is something like $f(\ell) * f(\ell)$ (i.e. autoconvolution)

.20 No!  front part d.f. is  backend d.f. is  (same)

I think in general the front & back end d.f.'s are the same. Is is like $\alpha - \int_0^\ell f(\ell)d\ell$ | $\alpha \equiv \int_0^\infty f(\ell)d\ell = 1$

So $= \int_\ell^\infty f(\ell)d\ell$ ∴ $\int_0^\infty f(\ell)d\ell = 1$

The d.f. for concat. is conv. of $h(\ell)$ w. $h(-\ell)$. i.e.

$f_{conv}(\ell) = \int_0^\ell h(x)\cdot h(\ell-x)\,dx$ which isn't exactly a convolution.

Well actually, since $h(\ell)=0$ for $\ell\le 0$, $f_{conv}(\ell) = \int_{-\infty}^{+\infty} h(x)h(\ell-x)dx$ — which is

.30 a kind of convolution. It is either an auto conv or a conv. of $h(x)$ w. $h(-x)$. — Different authors may use different definitions.

So: My impression is that when we concat 2 dfs like this, the d.f. of the resultant get wider, by say, $\sqrt{2}$ factor. (unless of course, we have a "selection" for shorter pgms).

There is also Q of: even if the GP & PPM give ≠ same D.f., is PPM more "efficient" than GP? — My impression is that PPM is rather "efficient" —

Also its use of Lsrch is efficient.

4TM

Intelligent PQ2. Intelligent Srch 2.0. ρ 16208 5.3.04: ∴ Tom Ward 781 646 3703 ← message; Jeff" Mari" 781 395 9034 message

.30 : It may be that w. 131.35, I'll not need a "phase 2"! Tho still, Phase 1 doesn't understand (fully) what "often" means. — So something has to be added to 131.35!

Hvr: — T. first "layer" is already Universal. — what does t. second layer add to it? — Well, it would seem that finding regys that PPM couldn't find, would be a v.g. thing!

NB. "Universality" means discovery in T<∞; But T can be very long! An already Universal induction System can be [redacted] improved by speeding it up.

Greg; Steve; Bill 492 7532; Bill: 617:492 7532 NO; Greg 734 3229 GTC pull

.07 Another task: Decide just what I want TM to lrn in, say 131.09 — .13: write pgm that

.08 could do it. Then make TSQ for TM to discover learn that pgm. → .15

SN in t. assoc pgs lbld 8/13/00 — A study for a TSQ in Algebra: When I want T. M to "know"

.0 "equals plus equals = equals equals" I want it to know not just phrase, but t. implications of it. — i.e. I want TM to "understand it" to some extent.

In t. case of 131.09 — .13, T. universal System must eventually, be able to do that! How can we speed up the discovery? We should be able to write a TSQ to enable / speed up that discovery

15. .08 → I think t. original problem was this: That TM learns to do problem types $a_i|_{i=1}^{k}$: T. problem descns are lbled by trainer using t. a_i symbols. This sort of thing would seem to be not too dfclt. Next we want TM to [redacted] lrn to recognize t. k prob. types w.o. t. trainers including t. a_i's. Let's look at t. details of both types of problem solns.

.19

.0 SN Perhaps t. soln. is that PPM doesn't have to be universal: that there will always be impt. kinds of regys that it cannot see / dev. Hvr, t. main regy discovery process is [redacted] (srch on pgms in a Univl. lang. — i.e. this is universal, T. [crossed out] did type of 131.09 — .13 [redacted] should be (must be) solvable as t. main problem type re. .07, .08; .15 .19....

So I think 131.09 — .13 is not a legit. Criticism of PPM's use in "guiding" srch on a universal set of instructions. .15 ff 13 of interest/import, hvr. I'd like to have a clear "soln" to it before I go on. (Tho this is not obvly nec'y.)

27 Going Back to .15: A couple of ways to solve t. first part!

1) We give TM $(Prob_1; a_1)$ (a_1 is index) after it solves it we give it $(Prob_2, a_2)$: It has to find a function that solves both probs. We then give $(Prob_3, a_3)$ & it has to find common soln. for all 3 ... etc.

30 ✓ 2) We give TM $(Prob_1, a_1)$: It finds soln. We then give TM $(Prob_2, a_2)$, but we are satisfied w. a "less than perfect soln." e.g. It may find F_1 that solves $Prob_1$ & F_2 that solves $Prob_2$, $(F_1 \neq F_2)$. However, its "total funct" now for any problem (old or new) is now t. δ function $prob_{1,2} \delta \left(\begin{smallmatrix} F_2 \\ F_1 \end{smallmatrix}\right)$. — This certainly / much better than no soln at all, but if $Prob_{1,2}$ have a funct that solves both, exactly, that would be better. At first, it would seem that TM has no way of knowing that a δ funct exists that solves both $Prob_{1,2}$. (Hvr, we may start TM out w. its assuming that all probs we give it have a common δ funct soln.)

It should be easy for TM to lrn to correlate a_i w. F_i — so it could get 100% correct answers,

4.29.04
4 PM

.00 : 130.40 : 130.37–.38 is impt. Learning to map probs to indices seq easy (fast) to best solns. Its much easier than lrng to map prob to ... needed to check solus.

Th. problem to be solved is finding a best dfty Σ funct — [illegible]

random method of finding params work? ...)

.05 → Hvr. a major weakness of PPM seems to be its inability to try params (or subfunctions) in dfrnt orders (?) Check to what extent this is true. (The Universal lang can do the sufns & this may deal w. this problem (tho perhaps not in best way).

.09 In general, if αa_i should be followed by β and a_i can have many (random) values: PPM has to learn each a_i case separately!

.10 In general, I certainly didn't expect PPM to be able to recognize all kinds of regys, but I'm disturbed that it can't deal w. this particularly impt regy type.

.12 t. regy of interest? That t. regy is internal to a seq. yet its needed response is after t. sequence. PPM only recognizes regys that are found for t. ending of t. seq

.13 Could we get t. recognition type that we want by > 1 "layer" of prods? → (see 133.05 for conclusion of following discussn)

[SN] T. Laws of Physics all disallow "action at a distance" — so all causality propagates via a chain of adjacent "disturbances" — yet physics is able to deal w. very complicated causal sequences.

.20 To what extent is (Koza's) GP (crossovers of function trees) able to deal w. ≈ .09 ff ?

e.g. We want to be able to categorize a set of seqs by how they begin. (Backwards: Context)

or There is a certain characteristic of t. first 10 tokens obtained by a certain function (strings → reals)

PPM couldn't discover much of this. Hvr. if we tried to find functions out first 10 tokens

.23 [5.1.04] PPM could help find appropriate ones.

Hvr, anything that can't be done by PPM can be done "at a hyer level" by a univl. lang. — just how to implement this in a good general way, is unclear. Perhaps t. UMC looks at t. first level codes & tries to find regularities! Or, UMC looks at Σ & tries to find regys — in addition to t. "normal" & a normal regressn. regys. Perhaps have UMC look at t. error PPM makes! After running a very long time looking for regys in PPM errors, on a very long corpus, t. UMC may get a "Universal correction" for PPM that signifty ↑ its accuracy. [10.4.05] 10k to 10^6

.30 On t. other hand, I expect that grammatical-like regys (noun, etc.) would ↑ PPM accuracy considerably. Would it be able to deal w. t. dfty of ≈ .09 ff. (say .12–.13).

A finite State Grammar mite help (ASMM). A possy: t. set of contexts that precedes either a particular token or a set of "backward contexts" (usually starting w. same token) — this set is a "state" candidate.

.35 Essentially t. problem of .09, .12–.13 is a categorizn problem.

→ A perhaps V.G. way to deal w. ↑ is another problem of capacity of PPM. T. problem of "improving t. pd" obtained by "PPM" becomes one of t. "low level problems" of t. system. By "improving t. pd" I mean to include cc as well as pc aspects of t. "pd": i.e. it is a "Gauzd pd" in which both pc & cc are considered: t. pc is a funct of cc as well as of t. corpus.

A Q is: Do I want to (need to) do this in phase 1?

4.29.04
4TM

0 : I mite be able to get .Y. I [redacted] out of existing BZZ pgms — & then try to predict it — compare w. BZZ, PPM, etc.

→ As for TSQ's: I could write a TSQ w. possl. solns. to problems, & see How many Bits were needed for each additional soln. T. values would be within a factor of V2 (i.e. t. 2 too small or K.l. t too large) — Good enuf! ⊙.

I would like to see if it could get solns better than Mine! Particularly if I use a long t.SQ., w. lots of problem solns — so that it mite find parts of solns that I wouldn't think of using!

---

06 Impt. [SN] on TSQ Design: I will teach it to solve a certain class of probs that have a common-ish soln. ~~Then learn to solve~~ Then " class, " and so on for several classes. Then it has to learn how what class a problem is in & apply appropriate ~~proper~~ soln.

It would seem that this kind of thing should be readily treatable by PPM. "The class of problem decider" (wholand) would be a / type prefix to [redacted] any attempt to solve a problem. → T. foggy discussn. suggests NO!.. This is Normally done in "Phase 2"

On second thot., I don't see how .06-11 could work! : (Deciding what class t. problem is, is a [redacted] "Meta problem":

3 The "Meta soln." (= Meta function): Looks at t. problem & outputs a p.d. on possl. solns: This distribn could be in

14 t. form of a narrow distribn on [redacted] "contexts" (these contexts being ones that are follwd by possl. prob. solns. sequences)

15 One way to approach this meta problem "soln": T. original corpus consists of (problem descn, soln) pairs. Unclear as to how PPM would be effective in this case. T. choice would depend on t. context at t. end of t. problem descn. ~~It's not clear that~~ I think this would not usually contain info needed by t. "Meta decision" of what problem class we have: Its t. kind of problem that Phase 2 is designed for.

18

~~We'd like~~ One way would be to try to find a function that maps from t. problem descn to a pt. on $\vec{L}$. (or distribn of pts on $\vec{L}$). A corpus ~~for~~ (ing such a funct, could ~~be~~ ~~the~~

20 prob descns & ~~[illegible]~~ paired w. pts on $\vec{L}$. This is a semi-discrete problem, since small errors in t. output are not impt. (i.e. if dist. width of t. $\vec{L}$ kernel, then errors $< 10'$ are acceptable.)

At t. beginning of TM's TSQ, t. the null context will eventually give a p.d. on $\vec{L}$ ~~of~~ over useful contins that are solns to problems. The Meta function will map to a "sub distribn" of this p.d. — "Narrow it down", so for each t. [redacted] "solns" in .15 correspond to pts on $\vec{L}$ that ~~are [illegible]~~ follow t. null context.

25 There are many possl. functions & sub-functions on t. problem space: Certain of them are useful to solve t. problems:

26 Certain are useful to solve t. meta problem.

---

27 In a prelimy training phase, t. problem-type index could be placed (by trainer) after t. problem. (It would not work if t. index came before t. problem descn ... this is a major criticism of this PPM method)

~~This would~~ If we then use problem, soln pairs as corpus, then inserting a new problem in, would end in the proper "index" of problem type & result in a good soln. tried, when t. problem (as context) was inserted into $\vec{L}$, to find how a soln may be generated. After this has been done

30 for many problems & problem types: we want to be able to insert ~~a~~ a problem w.o. it index.

— I think we then run into t. diffy of .15-.18! T. point is α & Λ is not at all close to $\alpha\Lambda$ in PPM.

Well, say we solve a large set of t. first order problems (w. or w.o. indexing ... t. indexing would speed up solns). After we've solved ~~[illegible]~~ them, we use that same $\vec{L}$, to develop, T. meta pgm that maps from first order problems into "soln. space", or simply from problem descn to indexes. While t. input subfuncts for t. Meta Soln, would be like those used to solve t. first order problems, the output functions,

37 would be much difrnt, unless we have given TM problems ~~of the~~ that are similar to t. Meta problem (at least in output form). [Remember Ob/op algebra: we have similarity of

38 probs to Meta probs ~~as~~ for Obs but not for ops.

8/13/00 — Dup. of an earlier page.

This will be an ordered list of problems, tasks, definitions, ...
--- toward the construction of an initial TSQ for TM.
167.24-.34
l, m, n are 32 bit random numbers
u, v, x, y, z are variables
":" separates examples.   "," separates data within an example
cond means "conditions for this problem"
imp   means "~~what~~ is implied by these conditions"
n=n e.g. 3=3, 7=7
cond x=n imp n=x
"[" and "]" are "metasymbols" (I guess I want [l+m] to be my numerical eval. of l+m.
l+m=[l+m] e.g. 4+5=9 --- learning Addition
[l+m]=l+m e.g. 9=4+5 --- Equality Commutes
cond x=y imp y=x Equality Commutes
l+m=[m+l] Addition Commutes
x+y=y+x Addition Commutes
x+m=m+x Addition Commutes
l+(m+n)=(l+m)+n Addition is Associative
cond x=m imp x+n=m+n --- if equals are added to equals the sums are equal.
cond x=y, u=v imp x+u=y+v --- as in previous example
l-m=[l-m] learning Subtraction
m-m=0 meaning of Zero
x-x=0
cond x=m imp x-n=m-n -- if equals subtracted from equals, remainders are equal

---

Note: TM may lrn $\left(\frac{m}{x}\right)\cdot x = m$; for many values of m, x : but for 1 value of x, (0), its not true : so its true w. prob'y 1 !

T. fore.g. are not / good Q.A. problems: e.g. cond x=n imp n=x (Q: cond x=n imp; A: n=x) — since cond x=n implies lots of things! like x=x, n=n, l=l, etc.

It seems better to give problems in which various principals are "discovered", in the sense that TM has developed mechanisms that make certain (operations/prob.solv. method) very likely. TM acts "as if" it knew that mult. was commutative. Also, we want TM to realize that all commutative functs are in imp. ways

One way to deal w. this: First Get TM to work probs as if it knew about commutativity of several Mathical functs. Then later get it to learn how the commonness of commutative functs is useful in certain kinds of Math problems. (It may be that this aspect of commutativity is an essentially diff'lt discovery that it took the Math community much time to discover ... in which case we may need to give TM heavy "Hints".

4.27.04

4TM

SM !!

129

.00: So 128.23–.40 expresses t. problem of computing t. kernal" as one way to do linear predn. — That [redacted] auto corr info of $\vec{Z}$ & soln. of Toeplitz Matrix will give t. M coifs of kernal in ~MN time (N is Orig'nal corpus length). Hvr. other means of N. Linear predn. (128.26 ff.)

It is not unreasonable to do a kernal w. several thousand pts. ("lags"). Hvr, we do want t. no. of ~~pts compute~~ coiffs (in kernal) to be << N because of overfitting diffy

To start trying this out, all we need to do is put a corpus in Lexical order! Perhaps we have to knew t. ~~To Know following each entry~~ I think it is the sequence of "following Entries" that we want t. auto corr of.

Note that we want t. auto corr. for each token, so we have R difnt. corpi.

SM Could we use cross corlns betw these R time series? Well look at some cross cors: see if they are "zero". "Zero" in sense of useless for predn. → .23

.11 SM 10 stocks: each has a token! Make corpus of tokens of stock that had max % (↓/↑) each day. Then ~ "Kelly bet" on pz's for each stock, every day (The bet will be a vector of upto 10 difrnt stocks — like a horse race! ☺). Actually, one may not be able to "kelly bet" because of unavailability of Leverage.

Picking t. 10 stocks to use, is not easy! One way: pick in same industry: likely to be correlated. Another: Use Currencies: correlated somewhat. Another Use Indexes DJI, SP500, ect.

Use various Spiders, Diamonds (≈ indexes)

Another trick! Each stock has 2 tokens: ② ① when it's (rises/falls) on that day. — So do time series ~~in some way~~ in same way,

But each day we have a pair of tokens in say Hi, low order.

IN this [redacted] SM. we have fixed set of tokens so we don't have to do "Escape" Evaln. → 161.00

.23 (.10) → In time series Curve fitting for t. kernal, we still have t. problem of "How many Coffs". T. must recount "soln" (which sounds reasonable), is to make apriori's based on t. past. (t. most recent past possl. for as close to present problem as possl.).

Hvr, in t. present case, t. apriori of t. kernal coifs is rather limited: they can't be < 0 or > 1. We could probably narrow it down further by looking at past kernals of content ~ to t. present one. ← They certainly shouldn't ↑ w. distance.

If we we assume coifs are ↓ w. distance, we have a relatively narrow distribn. Also t. ratios of previous adjacent coifs could be ~~other~~ in a rather narrow range. (Actually t. ratios of successive coifs that are actually used can be > 1 because t. previous coif could be wrong!)

If we assume t. coiffs ↓ w. distance, t. ~~[illegible]~~ coder cost of t. successive coifs ↓ a lot.

After Eyeballing t. set of coifs for successively ~~[illegible]~~ large no. of coifs considered, I can probably suggest some good forms for the kernal. — That ratio betw successive coifs varys slowly in a way that I can optimize. • so $e^{-|ax|-bx^2+c}$ may be good (only 3 params!) — only 2 if we want a normed kernal.

or $\frac{d}{ax^4+bx^2+c}$ or $\frac{a}{\cosh bx}$ ≈ $\frac{a}{e^{bx}+e^{-bx}}$

~~[illegible]~~

The question of apropo for nonlinear predn. (128.26) like x_{n-k} & x_{n+k} together, predict x_n. If they are both (0 / 1), this gets special wt. to predicting x_n. If they are difrnt they gets special wt to predict x_n. These may be t. only ~~[illegible]~~ symmetrical second (or third) order predns.

→ I'd like to be able to express t. kernal as $\sum a_i e^{-\lambda_i x}$ because it's easy to compute t. values. So try it w. 2, 4, 6, 8 values of i & optimize w. Marquardt method.

10 : A good way to estimate kernel: for each Token sequence, (0 or 1), do an "auto correl'n". It may fit to an exponential so take ~~[struck]~~ $\frac{\lambda}{2}$, since is the ~~[struck]~~ auto convolution of wtng function,

Say x_i is a randomish seq. of 0's & 1's at token pts.

$$\text{we want } \lambda, a \ni \sum_{i=-\infty}^{+\infty}\left(x_i - \sum_{j=i-1}^{-\infty} a x_j e^{-\frac{(i-j)}{\lambda}}\right)^2 = \text{min.}$$

I think $a \approx$ something like λ. I mean we want

Want the kernel to be normed $\ni \sum_{i=1}^{\infty} k_i = 1$; $k_0 \equiv \emptyset \leftarrow$ there's no prediction of $i=0$.

$$\sum_{j=-1}^{-\infty} a\, e^{\frac{j}{\lambda}} = 1$$

$$= \sum_{j=1}^{\infty} e^{-\frac{j}{\lambda}} = \frac{1}{a}$$

$$= \sum_{j=0}^{\infty} e^{-\frac{j}{\lambda}} = \frac{1}{a} - 1$$

$$\frac{1}{a} = \frac{1}{1-e^{-\frac{1}{\lambda}}} + 1 = \frac{2-e^{-\frac{1}{\lambda}}}{1-e^{-\frac{1}{\lambda}}}$$

$$a = \frac{1-e^{-\frac{1}{\lambda}}}{2-e^{-\frac{1}{\lambda}}}.$$

The idea is $x_i = 1$ for all i & normalized prediction to be 1. At any rate, for linear predn, the mean of predn. must = mean of predicted.

If x_i are random 1's w density d ($0 \le d \le 1$), then for large λ and a normed kernel, we will get a predn of d for the next x_i, which is the correct average. The larger λ is, the larger the SSZ & the closer we (usually) get to prediction, d. The goodness of predn is relatively insensitive to λ, but is best for $\lambda = \infty$. ~~[struck]~~ Here we assume the 1's are of ~ uniform density.

If they come in "bursts" (which is case of interest) then $\lambda < \infty$ may be better. — ~~unclear if coef a should ...~~ but unclear if kernel should be no longer Normed.

Say x_i comes in Gaussian shaped bursts & the burst area of density d_G:

Centers are (at random), one every $\frac{1}{d_G}$ (at) time pts. Say that $d_G \ll 1$ — so infrequent bursts. I guess λ would be about burst width, & maybe normed kernel.

Is it possible that there is a constant term in the predn.

---

23 Then, essentially, it's a problem in predn but one is allowed future values as well as past. I could try non-linear (cross products) terms. If I want to do linear predn, then cross correl'n [redacted] (≡ autocorr'n) has all needed info.

26 Cubic terms mite be v.g.! $\sum x_{i-1} \cdot x_i \cdot x_{i+1}$: If x_{i-1} & x_{i+1} both = 1 then very probably likely that $x_i = 1$. Also $x_{i-h} \cdot x_i \cdot x_{i+h}$ terms.

This could be a very interesting study in "Prediction" (Vilenkin?). We also have to scale the results so that the ~~meaning of coeff~~ wt. or coef of a term is a function of the term indices divided by SSZ of the ~~[struck]~~ original corpus.

The x_i is only 0 or 1, our predns are betw 0 & 1 (real) & are essentially pc's.

Treating the kernel discovery as a linear predn. problem, using auto corr'l'n info; It's a Toeplitz (P47: Num. Rec.) Matrix that has to be inverted & takes N^2 operations rather than N^3. P 433 has soln for common, less general form that I need — It could take $\ll N^2$ .... but I have to study this. Apparently takes $M \times N$ operations — M being no. of coefs & N the no. of data pts in the corpus. Actually, it takes about $M \times N$ operations to get the auto corr function! P430-433 of Num. Rec. (on "Power Spectrum Estimation by Max Ent (all poles) Method") seems directly relevant, since it seems to use the auto correl'n of the signal.

4TM

Something funny about t. kernel def.! Say w have a distribn of 1 at various lattice pts of width. For a fixed distance D. We slide the distribution past itself. For each integer displacement, we count —— No! It is not whatsoever!

Simulate a continuous d.f. . . .

Say we have a seq. of unit pulses / at lattice pts. (0, ±1, ±2) counting in bursts usually, but also some random background & weaker bursts & broader bursts. To predict (0 or 1) at a point, using to knowledge of all other pts: (1) What would be a good kernel? Say we use Max likelihood — we also have to predict zeros.

→ Actually, t. real problem is to assign to each token & to estimate pc for product of pc's assigned to all tokens. One way to do this is to try to do a kernel for 1 token at a time! to get max product of "Yes" & "No" pairs by t. kernel. While back I discussed the construction of an approx to a kernel (≈ 17.32 ff) Minim. of $\ln\left(\frac{\text{estimated}}{\text{true}}\right)$ but this assumed I had a "non-parametric" model of t. kernel to "smooth" (& approximate) — so it doesn't look so useful now. . .

But t. idea of expressing & t. kernel as t. sum of exponentials (Laplace xfm) still sounds Good — but t. error criterion will be different — simply Max product pc. ≡ Max likelihood. I can either do each token individually or do them all at once. Calculate whether it would be much faster to do them individually. — One way is — Assuming all tokens have same kernel — One could do one token (Type) by itself, then use that kernel as an initial approxn. for doing them all together.

I can just use a 1 sided kernel to get t. kernel params. Then use both sides for actual pairs. Perhaps try each side individually to make sure they are about the same. Eyeball t. data to estimate good starting params. The optim. program is pretty much in dispute of t. Gaussian. I can vary both t. roots & t. exponents to used in t. "Laplace xfm". — So start maybe w. 2 exponentials (4 params) then go 3 exps then 4, etc.

A major problem is to make sure t. kernel is ≥ 0 for all values.

It is possible that Muple has in it's standard opt Nonlinear Optimization pkg. But porting t. data to it would be non-trivial.

Num. Recipes desribs "Marquardt method" pp 523–528; ← Try Google, (& also see Books on Math Modeling). IN PS folder: dumpster-diving "Laird method" 3/25/04 is in DJVU (1977) 670K. Maybe get Alex to rip it. . . Linux does DjVu files.

Around 3/25/04 "EM algorithm" There was 6 files 3/25/04 on EM algorithm which is usually used for optimization of Max likelihood . . . . but w. missing data"

EM.PS gives a good picture on pp 1, 2, 3. The method itself is "obvious", but the proof of always converging, is not. First page 3 pp: It's not clear how to apply EM to the problems of fitting a 2 pole kernel model.

The various EM papers may tell how to apply the idea to very many heterogeneous problems — possibly suggesting how to do "t. kernel" (which may be a common problem).

4/22/04
4TM

Get good pc's/~~for~~ continuus of T. TSQ! [inserted above: solns of probs that are]

[SN] In Induction/TSQ aquisition: We have ① a universal lang [inserted: L_0] to express program solns.
② an initial pd/on [inserted: P_0] sequences of statements in that lang. This initial pd/[inserted: P_0] is not a universal dist, ~~[illegible]~~ but it enables a certain amt. of lrng.
For P_0: 3 degrees of sophistication! ⓐ symbols of L_0 each have/~~had~~ [inserted: known] pc's so pc of a cand is something like exp(lengthof cand term). ⓑ like that but pc's change; accumulating pc via Laplace ... assume bern seq. for Tokens of lang ⓒ PPM-like dist. on Tokens.

4/23/04 Q_1: For a universal lang, is it necy to have definitions "or equiv."?

By Q_2: At what pt. can we ~~[illegible]~~ give t. system t. problem of improving P_0 a/o L_0.? — To some extent P_0 improves itself in response to TSQ.

→ If defns. are possl., it is possl. for L_0 to be effectively modified to any lang (cound. or not)

Q_2: Depends on L_0, P_0 & TSQ.

In "Phase 1" of ALPHA, we give [QA] problems: T. system wants to find a single funct to solve t. whole set: At first Deterministic, Then probabilistic. In either case, past solns (or subsets of "T. present" TSQ) are used to give a PD. for Lsrch on t. next "Corpus" (& old corpus + soln to new augmented corpus). The Corpus is a seq. of solns to "t. same problem" — or to "aspects of t. same problem". It seems to fit a ~~[illegible]~~ GA prob-soln. & i. a PPM guided Lsrch.

(SN) Re: GP, GA: There is something unreasonable about the "Monte Carlo srch" Taking ~~expected~~ time, N where N is no of differt cands. N → ∞, yet t. system seems to work! — How? Well, One way: we pick cand i w. P_i where P_i is pc of cand's being "correct/soln". [inserted: (verifiable, testable)] We decide on time limit, say $10^6 = d_1$.
We do M trials, each lasting T secs., max. We then repeat after $T \leftarrow 2T$, etc.
If t. soln. has $P_j > 10^{-6}$ and it takes time $= cc_j$ to generate & test it, it will take total of $\sim cc_j \cdot 10^6$ time to find it. — which is much worse than Lsrch, if $P_j \gg 10^{-6}$.
Lsrch got $(2)\frac{cc_j}{P_j}$ time for srch.

In QA Phase I: just how did I go about representing probabilistic Models?
I think I wrote a lot about this: perhaps including summaries.
For Phase 2 I mainly needed PD's on PSMs, problem pairs — & this was a 3 param distribution — [inserted: Zaweh] mean, var. & ~~[illegible]~~ moment; so 0, 1 & 2 moments.

Before Going on to t. Details of Phase 1 & 2, write a good usable summary of just how T2T a ll & rand are to work!

I don't quite remember just how t. kernel was computed!
Somewords: ~~103.05 ff~~ : ~~[illegible]~~ 99.00ff early ver...
~~2~~ 1 pt. unclear | ~~a~~ not bad!
107.20 ff isn't 100% clear, but I think we can find a kernel for each token, indep of others: They are probably all same shape, so we can average them. (To check on whether t. shapes are indeed t. same),

4TM

00 : 124.40: perhaps I can update in T2T when T interval gets sufficiently > Δ (Time needed to update)
So: I guess that I have to ppm in enough detail to write out a more detailed desc of all
of Lsrch — for T2T Lsrch & for ll & Random to some extent.
It might be well to do T≥3T Lsrch! I've forgotten the details of the proof of optimality!
This does have a slight advantage over T≤2T Lsrch.

I really can't update more freely than after each prob. soln. for T2T Inv probs. — Since there's nothing to update with!
Hvr., for OZ problems, perhaps since I can have already obtained some "soln." (I can update before "final soln.") at any pt. in the soln.
This → is essentially very ≈ to GA. Since I "ultimately" (for phase 2) want
to solve OZ probs, This ≈ GA is an impt. direction to go in.

10 The eq of 124.33 $m = \sqrt{\frac{2a}{\delta}}$ assumes updating & soln. time. In OZ probs,
This idea has to be modified! one way: Compare the pc's of hy G cands:
before & after an update. I.E. after an update, find the pc's of various hy G cands that
were obtained before the update. Since time to soln is ∝ $\frac{1}{pc}$ This is directly translatable
14 into "δ". Actually, updating is designed to ↑ pc's of previous hy G cands. It should do this w.o. overfitting. → (17)
Work out the operation of Lsrch for OZ problems. Certainly (10) seems to be an impt part of it
perhaps (14) can be used to estimate δ more accurately!

17: (14) → The way it works: After every update, we recompute δ (because it takes little time). After an update of
m cands, we compute new pc's of each of the m cands w.r.t. the corpus & compare these pc's to
the values obtained before that update. The total difference in $\frac{1}{pc_i}$ will be $m(m-1)\delta$ (note factor of 2 vs
w.r.t. the $\delta\frac{m(m-1)}{2}$ of 124.31.) This "δ" computation gives us feedback on → (For we may want to delete a cand from the corpus before finding its pc wrt the corpus)
how well the system is "lrng"! ← Very impt! δ is a very incisive comment about Efficacy of the TSQ!
This "δ" was very small in the English corpus used in the original BZW 1994 paper.
The δ & the corpus length is seemed to → 0 for corpus of ~ 150M chars.
Make δ estimates for the 100M char corpus.

Can we use the (very fast) BZ2 pgm to Estimate δ in TM! (1) — We can do this
because to Estimate δ, we don't need the pc's of individual Tokens: only the products
of those pc's — which BZ2 gives quickly.

From the 1994 B&W paper. Table 2, p14:

Corpus 1k → 4k symbols. $\frac{.52}{3k} = .17$ m bits/symbol (?).

4.35 bits/char. 3.83 Δ = .52

But $\frac{.52}{(3k)^2} = .06\,\mu$ per symbol.

B&W Data		Δ
1k	4.35	
		.52
4k	3.83	
		.44
16k	3.39	
		.41
64k	2.98	
		.32
256k	2.65	
		.22
1M	2.43	

Interpreting the B&W data is not so simple!

Anyway, what I can do is write a seq. of problems & solns, & see if the PPM gives pc's
to new probs in an acceptable way. — Does it pick up the sub-functions that I expect it to, etc.
This, of course, takes much less time than letting TM actually srch for solns. — Also it's
much easier to pgm! I could get a useful approx. from a few BZ2!
This last is of interest! Write some seqs of problem solns & see if BZ2 can

4TM $\frac{m^2-m}{2} + m =$ ⋮ $3+2+1=6$ $\frac{m^2+m}{2}$

$\frac{m^2+1}{2}$ $\frac{3\cdot 4}{2}$

00: 123.40 Away to think about it: Consider a seq of probs. Must have been done w. updating after each problem. This is "system 0" ($\equiv S_0$). How much worse/better is it to update every m problems? (Time needed to update is same constant in all cases).

update 0 0 0 0 0 ... 0 update
1 2 3 4 5 m
prog prob.

S_m system that does this.

No: $\frac{(m-1)(m-1+1)}{2}\cdot\delta$

The update on any problem causes subseq problems to be worked slower by time δ.

At first problem after update, both systems (S_0, S_m) are same.

After second " " S_m is behind by δ

Third " " " $\delta + 2\delta$

fourth " " " $\delta + 2\delta + 3\delta$

m^{th} " $\delta(1+2+3\cdots m-1) = \delta\frac{(m-1)(m-2)}{2}$

show by ind $\frac{m(m-1)}{2}$

(impt!) If (you) don't understand why of this "sq law": ! Oh! The idea is that the k^{th} problem is deficient by not having $(k-1)$ updates of the previous problems

Say Δ is time needed to update. When m is such that $\Delta = \delta\frac{(m-1)(m-2)}{2}$ then

S_m is as fast as S_0.

At end of m probs we update so total cost per problems is $\frac{(m-1)(m-2)\delta + \Delta}{m}$ ← This is cost of last m probs. a S_0 that has zero cost for updates

To find m ≥ this is min. $\frac{m-1}{m}\frac{(m-2)}{2}\delta + \frac{\Delta}{m} = \min$. $\frac{m^2-3m+2}{m}\frac{\delta}{2} + \frac{\Delta}{m}$

$x = m-1$ $m = x+1$ $\frac{x}{x+1}\frac{\delta}{2} + \frac{\Delta}{x+1}$ min. $\left(m-3+\frac{2}{m}\right)\frac{\delta}{2} + \frac{\Delta}{m}$

$1 - \frac{1}{m} + m\cdot\frac{\delta}{2} - \frac{\delta}{2} + \frac{\Delta}{m} = \min$ $\frac{m\delta}{2} + \frac{\delta+\Delta}{m}$ min

$\left(1-\frac{1}{m}\right)(m-2)\frac{\delta}{2} + \frac{\Delta}{m} = \left[(m-2) - 1 + \frac{2}{m}\right]\frac{\delta}{2} + \frac{\Delta}{m}$ min.

$m + \frac{2}{m}$

$\frac{d}{dm}\left(\frac{m\delta}{2} + \frac{\delta}{m} + \frac{\Delta}{m}\right) = \frac{\delta}{2} - \frac{\delta+\Delta}{m^2} = 0$ $m^2\delta = 2(\delta+\Delta)$

$m^2 = 2\left(1+\frac{\Delta}{\delta}\right)$

$m = \sqrt{2\left(1+\frac{\Delta}{\delta}\right)}$ looks ok! I checked it.

I can know Δ but δ will be quite uncertain. I can determine δ empirically by working a seq. of m problems; then work at "pt 2".

$m = 2$ if $\Delta = \delta$.

from pt. 2 we work a bunch of probs. — we see how low it takes. I go back to "pt 2", update for previous m probs, then work the same batch of problems & see how much time urgent.

26 Or, just after an update, we redo several problems that occurred before last update. We can pick an initial m, then do .26 periodically to estimate δ. Since we only redo, say, 10% or 20% of k problems for .26, this will not be much of a penalty for determination of δ.

$\left(\delta\frac{m(m-1)}{2} + \Delta\right)/m = \min$ → $\frac{\delta}{2}\frac{m-1}{m} + \frac{\Delta}{m}$: $\frac{1}{dm} = \frac{\delta}{2} - \frac{\Delta}{m^2} = 0$ $m = \sqrt{\frac{2\Delta}{\delta}}$

So $m = \sqrt{\frac{2\Delta}{\delta}}$ see 125.00 for truncation of this eq. for 02 probs.

see 138.13 for a cheap way to do updates.

Anyway from (23.03, .13, .14; $\Delta \equiv N_L \cdot R \cdot n_2$ (10 to 20). This could be 200 N_L: N_L is no. of tokens in t. entire set of problems solved. T. time to solve a problem could be 10^9 tokens.

Here "solve" is not proper term for 02 probs. 100 problem w. 10 tokens each is $N_L = 1K$;

So $\Delta = 200k$: which is << t. 10^9 to solve a problem. So maybe update every problem?

At first, it should take < 10^6 to solve a problem so may not be feasible.

In T2T I was thinking of updating more frequently than just "before problems". → spec 125.00

Mandas
Marlowe
Hauld
Fay

:00 : Trouble. 122.37 : If a cand is [illegible] length D_c and the kernel of width W. — It takes ~ $W \cdot D_c$ operations for "correct" $\vec{L}$ profile with addition (or deletion) of the cand.

03 On the other hand, a complete update of L's pc's requires ~ $R \cdot N_L \cdot n_e$ [No. of "Ledeca coeff" used.] operations.

Maybe only $N_L\, n_e$: We move along the L corpus 1 by 1; each time we do a token, we update only 1 of the R Kernelizations of the corpus.

T. update algm is $Y_{n+1} = (1-\alpha) Y_n + \alpha x_n$. x_n = 0 or 1. (perhaps a conformant parameter ∴ a shift.

So $1-\alpha Y_n = Y_n - \text{shift}\, Y_n$ ~~[illegible]~~ So $Y_{n+1} = Y_n - \text{shift}\, Y_n + (0 \text{ or a constant})$

So we have to update all R Y^j's, but it could be a fast update.

Shift Y_n ~~[illegible]~~ seems to be a fast inst — 2 or 3 clocks in 486 — maybe faster Pentium.

) So we do $Y^j_{n+1} = Y^j_n - \text{shift}\, Y^j_n$ [shift R] for all R components. We then ~~[illegible]~~

.11 add a ~~[illegible]~~ small constant to one of the R components: (then $n = n+1$.)

So the big time thing is ~~[illegible]~~ $Y_n -$ ~~[illegible]~~ shift. This is unit be done in ll. in Pent II III or IV.

.13 So It is $N_L \cdot R \cdot n_e$ ["no. of Lap. coeffs"] : But this maybe t. no. of clocks or no. of clocks × 8 (or ×16 for possibly

.14 newer 64 bit or Pent IV or whatever) . Say R = 10 (min); n_e = ?? 20 ? ⊕ : so $200\, N_L$.

Say we are keeping only t. top 10% of cands — still, N_L can be large. It corresponds to GA "population size" is $N_L = 1000$; ∴ $R\, N_L \cdot n_e \approx$ 200k clocks for one update of $\vec{L}$'s pc's.

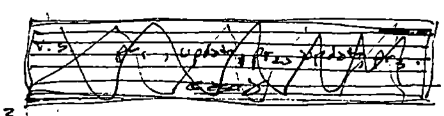
N.B. ~ → .. t. "small constant" can be ∝ t. Goc of the cand at that pt., w.o. appreciable ↑ in cc of t. update !

0 SN "Inner Loops: "Book "R250" pp 235-236. He has a 10 clock algm for generating random nos. which he says is Very Good, fast.

So: See how much time update takes & compare to t. amount of Benefit we get from it. Both expressed in common units of "Time lost or saved". This is a classic problem about Scheduling Problems v.s. Meta problems! — But more realistic ~~[illegible]~~ better param knowledge [illegible] etc.

So say update takes Time t. If we update first, subsequent problems will be solved [illegible] a fraction of time δ sooner (T → T(1−δ)). [124.00 so t vas this] ↑ is a function of t. mean Goc of t. problems in t. update.

So pr_1, update, $prob_2$, prob3 v.s. pr_1, pr_2, update [1,2,3], prob3.
↑ case α (β case)



V.s. pr_1, update pr_2, update, pr_3. ← γ

3 In 2 cases, compare times for probs 2 & ~~[illegible]~~ times for prob 3:

α v.s. β In α prob2 is solved faster, ~~[illegible]~~

But in β t. update includes ~~[illegible]~~ pr2 in addition, so pr3 will be solved faster.

In γ, the 3 problems are each solved as fast as possible using t. best info possl., but where cc of an extra update. [has been using t.]

I need [illegible] model telling how much an update & soln. time as a function of what's updated by

Consider only first order models: So ~~[illegible]~~ $prob_1$, $prob_2$, update [β] and prob update, $prob_2$ update [γ]

both solve $prob_3$ in same time. This disregards fact that in γ, t. soln to $prob_1$ is better than that of β, so t. second update of γ uses one better soln. than t. update of β. In spite of this slide betterment of t. prob3 soln. by γ, it costs a whole extra update.

So, assume an update at a particular point summarizes all update info up to that pt.

4TM

.01 : in advance by convolving the kernel w. $\vec{L}$, R times

Say N_L is # length of $\vec{L}$'s corpus. Then it takes T times $N_L \cdot R$ to convolve the kernel w. $\vec{L}$, R times.

I'm not so sure of this "c" factor! perhaps mult by c, the width of the kernel

On the other hand using T2T as in 121.28 we take about N^{cands} (= no. of cands tossed) computations of pc's of Nodes. Each node may take time $N_L \cdot c$ to compute: I don't know what c's value is, at all! — It is < 1, but how much less ... I just don't know or have any good ideas! So total time $\sim N^{cands} \cdot N_L \cdot c = c \cdot N^{cands} \cdot N_L$.

So compare $N_L \cdot R$ to $N_L \cdot c \cdot N^{cands}$ certainly certainly $c \cdot N^{cands} > R$.

So it would seem best to calculate all of the pc's in advance via .00 – .02.

A better comparison is $(.00) \times c \equiv N_L \cdot R \cdot c$ Vs $c \cdot N^{cands} \cdot N_L$ or R v.s. N^{cands}.

R is still $<< N^{cands}$. — So .00 – .02 looks best.

The analysis of 121.28 of T2T still is relevant & makes T2T look up. We started out by computing $\vec{L}$, then we computed the kernel, then we got the "Laplacean Xform" of the kernel (via .117.00 – 119.03). Then we get R pc values for each point along $\vec{L}$ by convolving the approximate kernel w. $\vec{L}$.

We can now do T2T or (I don't know) Lsrch.

T2T seems only a factor of 2 slower than ((our Random)) yet requires Much less RAM.

Random Lsrch has speedup in that pc's need not be calculated, & perhaps fast (Quick and dirty) Pseudorandom No. Gen can be used. Nvr, it still seems to use a lot of RAM.

If we had the RAM available. (It needs $\sim N^{cands} \cdot D$ (Length of cand = "depth") of RAM), would it be faster than T=2T? Well even if we had the memory available, the time $N_L \cdot R \cdot c$ (if it is, indeed a heavy factor) Is PERHAPS much smaller than N^{cand} (# of Nodes that we need to compute).

In .01 instead of multiplying by C, perhaps mult by (N·c)!? More linearly: Mult the $N_L \cdot R$ of .01 by n_{bits} which is no. of bits we use in approximating the kernel by wtd exponentials.

Is $N^{cand} \sim (pc^{soln})^{-1}$? So up to 10^{10}? On the other hand N_L will probably be $< 10^6$

So it looks like .00 – .03 should take relatively little time — Negligible.

The loop is O.K. if N_L is $<< N^{cand}$; but for a single T2T Run to soln.; but If we want to modify the P.D. in view of problem solns, It seems like a different story!

On the other hand, a single set of cands w. assoc Gores & a known kernel, affects the various pc's in a simple way; Can I implement it in an inexpensive way?

Removing a Cand from the L dictionary has a similar effect — recalculate them that way.

Can we use the Laplace xform approx to cheaply implement the addition/deletion?

I think we just consider the Gate seq. $\Delta\vec{L}$ given by the new card.

Filter it thru the Lap. xformed kernel & add it to the old $\vec{L}$! How much Time? (<<)?

7/20/04

4 Th — Bill's birthday. to Trobridge, Arlington Tuesday 1 wk 630.

20 : Speech Recognition: To discover how Humans recognize phonemes; See what kinds of errors they make (say w. 1 of noise: Gaussian Noise & random phoneme noise)
Also do error study for synthetically constructed phonemes v.s. ones obtained from Humans.

Refs to QNX
1) Do QNIX on Google.
2) QNX Real time platform,
Search Phoenix Developer Consortium
Google: QNX
QNX Momentics
NC Download 6.2.1
Z Non commercial.

03:120.40! **T2T:** I need (lupe) pc's for each of t. Tokens during cand construction. How much precision needed? (not much).

One (long) way: Get Z & smooth w. kernel, R times to get pc vectors.

A very approx. way: at each pt on L, we move along until we find various tokens. -- (do both w. 1, 2, 6 or cer). I have to do this until I get to pt. at which we have pc of least likely Token. — which is an awful lot of testing! T. no. of tests

10 (depends on) ∝ to corpus size. Also width of kernel. (Normed kernel ∝ to constant width).

We can estimate t. amt. of time spent on such activity. Essentially we have a Bern seq. of choices & t. freq. of each choice type is its pc.

$\sum_{i=1}^{R} \frac{c\,p_i}{p_i} = c \cdot R$. $\frac{c}{p_i}$ is time spent on t. i^{th} most likely choice

t. constant, c will perhaps be ∝ corpus size. (& unnormed width of kernel).
Hwr, this → is t. corpus that generates L, not a population of cands.

A possibly very useful cc. reducer! That one gets all of t. pc's on a single run along L.
Perhaps this amounts to saying each node computa (R potential pc's been possibly computed) takes on t. average, t. same amt. of time. T. no. of nodes is about equal to t. total no. of cands created (at a particular T "c B").

20 So we create a cand, for each node developed. For cands w. D (depth) tokens deriving them, that's a big saving.

T. amount of memory of "system states" amounts at most to D nodes worth.

This technique of doing T2T may give us a factor of D in speed w.o.t. enormous amount of memory needed by || or Random search. (The on second thought, it may be possible to use this same trick to ↓ amt of RAM needed for both || & Random search)

As a result, random & || srch may be better by T ← 2T by a factor of ~2, but otherwise, use about same amt. of RAM. No, || & rand do have to store much more than T2T.

28 → HVR: (NB) In T2T, one must store all of t. states/nodes that have not yet been completely developed. In depth-first srch, One explores t. tree in a "stick of left" manner,

30 so not many nodes are "In process" at any one time. — however > t. depth of t. branch being worked on. See the old pgm for L srch for INV problem in my ~1985 "optimum Sequential Srch".

I may want to write up this analysis of "optimum T2T": That it's almost as fast as ||, rand, w.o. much RAM needed, seems like a "BIG Deal"!

A perhaps impt. Q: Do t. four. sugts apply to Lisp type UMCs, in which one normally finishes a derv of a cand, before testing it?

T. Question that brot on this latest development (& surprise!) was: how much time it took to calculate pc's "on t. fly", rather than compute them all

4/19/04

TM

20: : An Alternative approach to Optimization (v.s. Lsrch Phases) is GA guided by a PPM. We can keep population at [redacted] larger constant size by deleting cands, when better cands are found.
On second thot it seems that I have to do Lsrch (of one of 3 types) to do GA, anyway! —
So GA is not an Alternative to (PPM Lsrch): its just another (training & Analysis, & comparison) & part of Lsrch Development.

Presumably, after a sequence of problems of suff size, has been solved, the guiding Pd is changed. ~~Whatever~~ Just how this should be done, is unclear...
One could continuously add cands to the base of the L corpus, but computing pc's becomes harder —
I.e. one can't smooth the $\vec{L}$ distribution w. the kernel — which normally enables very fast operation of the system.

Maybe 2 phases of operation: At first, befor we have any good cands, we use a constant L distribution. When we accumulate enuf better cands, we add them to the L corpus & recompute the L distribn. In general, we may want to do this — but we only add a batch of cands to the L corpus (i.e. recompute L), whenever the set of new cands really looks rather good, i.e. has lots of members above the 90 percentile of Gore.

In general, one does do Lsrch "to help GA", one uses Lsrch instead of GA.
Hvr, to be competitive, the L distribution has to be continuously or periodically updated —
(i.e. Learning during Lsrch).

If we use a TSQ, we can update L. d.f. betw. problems or after "every several" problems.

-3: 119.37 : Another possl. advantage of Random Lsrch is that the $\vec{L}$ d.f. can be continuously updated.
Its not nec to smooth $\vec{L}$ w. the kernel. We assume, hvr, that the kernel's constant — or that it is Updated infreqly.

A factor that the forg. doesn't explicitly consider is, in Optzn problems, The cands are not "solns" or "nonsolns" — they get G. values (= fitness). One way of dealing w. this is to include in the corpus of $\vec{L}$, only the top 10% (20 — or whatever) percentile of the ~~G~~ G values, ~~[illegible]~~
~~[illegible]~~ So the G Threshold for inclusion ↑ as we "climb the hill". This would be nice to do w. (.23) (random Lsrch)

In T2T Lsrch for OZ problems; we do a "T" run until we get some outputs then we modify $\vec{L}$ for ~~[illegible]~~ T < 2T.

→ To start out, our initial L could be constructed w. corpus of "Hand-solved" problems!
I had originally thot of having TM solve a bunch of problems using some fixed approach (I thot of Z(4), but in fact Z(4) doesn't change the pc's w. new solns, rapidly enuf to be much ~~better~~ than constant.
~~After enuf probs hav~~ After TM has solved enuf probs this way, we use PPM for guiding pd.

→ Drew up outline for each of the 3 Lsrch systems giving bits to how each part works.
Discuss how each is used: what kind of TSQ's it needs, what kinds of GA probs it can solve.

4/18/04

4 Tdy

.00 : 118.40 : Another way to get min $(\ln x - \ln y)^2$ is to use non-linear optimization using that second order method I invented; It is prob'y identical to ... Marquardt method.

.03 — I think it converges rapidly.

So anyway, say I have a good way to get pc's of Tokens. I can use it to realize
(Dcl) T2T 1) || Lsrch, 2) Random Lsrch 3) T↔2T Lsrch. ← call it T2T Lsrch

For T2T Lsrch, [all I need is 2·R params that "partly" describe the distrib: i.e. the 2R pts of equal percentile intervals.] ← NO! : I need the pc's of each of R Tokens (or better still, their logs — that I can get approx'ly using that trick from "floating pt": Actually keeping track of "where one is" in T2T isn't so easy! One way would be: that every fixed ΔT interval, one checks to see if $\sum \ln pc > \ln \leq \Delta T$:

.12 I might want to use pseudo floating pt! I use 2 registers for each no. The "characteristic" is multiplied to get "characteristic" of product. The mantissas of 2 nos. are added.
.16 We look at char of product. If it has a leading zero, we shift and decrement mantissa by 1.

.12 → .14 looks like a very fast way to keep track of products of pc's! but I'm not sure integer mult is any faster than floating pt. mult! Its floating pt. addition that's slow.

When I'm doing $n \leftarrow n+1$, I just keep track of carrys, As soon as one occurs, I inc. mantissa and shift characteristic. So I always have it in 8 bits., & I always know approx'ly (whether good enough) its $\log_2$.

.518 to Get $\log_2$ approx'ly: The characteristic is always below .5 & 1. To change this to t. $\log_2$, we need to subtract a certain amt, then mult by a constant. Since t. accuracy isn't hy, we t. mult part could be done by 1 or 2 shifts & additions/subtractions.

Also, we may be able to do approx. division between nos. in this ≈ floating pt. We do a few "shifts & subtracts". On basis of signs of results, we can get approx. quotient. T. pgm. can be written "in line" to ↓ time cost.

T. stuff from 117.00 to here is all v.g. & eventually quite useful! — Hvr. earlier discussions to 94.00 & earlier are also Impt. — T. cc of compu. is as impt. as t. accuracy of t. pc being calculated ... So I will eventually have to do that bit by bit analysis. But now, I want some f.b. on whether "TM" can help with TSQ's

.0 to make Phase 1 feasible — So I want to pgm it, use T2T & try it on a TSQ.

Also perhaps, try it on GA problems that have been done using standard GA methods
I want to know how many trials are needed for soln. of GA probs u.s. normal means ^

Going over this a bit! T. advantage of || & Random Lsrch is t. factor of N, in Time.
N is no of tokens in soln. T. Disadvantage is need for much Memory. Hvr, there is also a factor of N reduction in Memory: T. result may be that one can do it w. not so much RAM, Maybe

.37 ≪ GBy. Advantage of Random over || is that maybe less computation to get choices of Tokens. → (120.23)

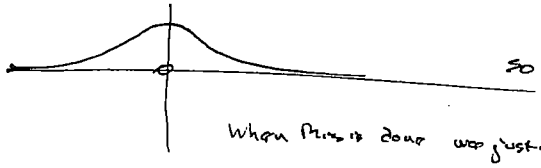
Advantage of T2T is (...) much less Ram, & perhaps much less keeping track of things (just simple depth-first tree Srch).

TIM

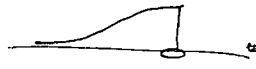
0:117.40: Probably a good idea to augment the 2-3 tree code with insertion. Then when we've saved it works, transfer it completely or mostly to Assembly code.

02: 117.05: The Wts (Cj of the kernel) can be found by curve fitting, using a "error² of ln(ψ)" as Gore — but look into the Gore carefully! The reason we want to minimize error in ln(kernel values) is that these are closely related to pc's & it's the product of pc's that we want to maximize — or the sum of ln pc's that we want to maximize.

It may be easier to divide the approx. kernel into 2 parts, because it is symmetric:

10  so just try to append:

When this is done we just get final kernel by running over the corpus; then

 run  over the corpus backwards.

I think I had a way of doing non-ms. error minimization by Successive Approx.

The thing that varied in successive trials was the wt. given to each approx pt.

So we did by min ms error, but need to tighten so that it becomes equiv. to ms of error in ln.

Say x is a data pt. & y is a pt we want to approximate x. we want $\left(\ln \frac{y}{x}\right)^2$ to be min.

20 We assign a wt, w, so $w\cdot(x-y)^2 = \left(\ln \frac{y}{x}\right)^2$.

First we try to get $(x-y)^2$ to be rather small, using wts = 1. Next time,

21 $wt_{n+1} = w_{n+1} = \left(\frac{\ln y_n - \ln x}{x - y_n}\right)^2$ ~~then continue doing this~~ from the new set of y's;

we recompute wts in accord w. 21.

If this doesn't converge: 2 possible ways: (1) do slower convergence, so

$w = \left(\frac{\ln y - \ln x}{y - x}\right)^{2\cdot\gamma}$ in which γ is <1. as γ approaches 0, w approaches 1.

(2) Somehow, do it backward: i.e. in solving $x = f(x)$ (for known $f(\cdot)$)

we can try $x_n = f(x_{n-1})$ successive approx. If it diverges, then its likely

30 that $x_n = f^{-1}(x_{n-1})$ will converge. How we would do this w. 21

31 in .21 since y_n is a function of w_n, we can write $w_{n+1} = H(w_n)$.

Here the inverse of H is quite complicated.

Another difficulty: It may be that using .21, w converges at some x pts but diverges at others. The inverting eqn .31 would then simply

Since the $(\ln x - \ln y)^2$ error criterion seems to emphasize behavior for small x,

for initial w, perhaps make $w \tilde{\propto} \frac{1}{x}$ (Note in previous case x is always >0)

we don't ever want $y \le 0$ (so $\ln y$ is complex)

.30 ∴ At present time: It would be possl. to convolute L w. kernel in time ∼T or T ln T.
∼ Time ∼ T might be done by simulating kernel by its Laplace xfm, the do $x_i = \sum \alpha_j y_i + (1-\alpha_j) y_{i-1}$
More exactly, kernel smooth $\vec{L} \equiv \vec{X}$, $x_i = \alpha_j L_i + L_{i-1}(1-\alpha_j)$
No: First we get $x_i^j = \alpha_j L_i + (1-\alpha_j) L_{i-1}$ for i=1 to T. for various α_j values
.04 Then kernel k smoothed L_i is $\sum c_j x_i^j$. Here c_j is det. by the Lap xfm of
.05 the kernel, K. We may be able to use the same kernel (no recomputn. of kernel) for a slowly changing $\vec{L}$, → (118.02)
which would be easier if we keep T constant, by deleting a corpus member every time we add one (corpus member).

.10 So If every workout we get a bunch of smoothed L_i: each is a vector (for each) having a no. of components ≡ the no. of Tokens.

.11 This would facilitate T → 2T Lsearch & ILsearch, bit output every 10 machine cycles! → (.32)

.12 The random Lsrch needs a fast random no. generator (inner loops has one!), say random nos between 0 & 1, & we make boundaries, and a fast sorter of the random nos. (maybe 2-3 tree) "Inner loops" has a fast random no. generator.

On second thot .12 doesn't seem relevant! Use .00–.11 to get R components of a pc vector (there are R diff Tokens). If S = sum of R components, we pick random no. on (0,S) & determine which interval it falls in: One way say Z is random no 0 ≤ Z ≤ S.
.20 We do $\sum_{i=1}^{r\,(r \le R)}$ Component$_i$ ≥ Z we pick min r > this is true, this takes on the average, $\frac{R}{2}$ additions — which may not be bad! Trouble is, we have to do $\sum_{i=1}^{R}$ compts to get S! So we do it $1\frac{1}{2}$ times! On the other hand, at each "i" point we may compute a "S" anyway, to use for normalization. But usually R additions will take little time: we have to load a register repeatedly w. new components.

use Xa, Xb: add xa ← xb [xc + xd]   This may not take so much time.
Incr xd
If xd < R goto

I may not need v.g. random nos. — try some very fast (not very good) ones —
.30 See if using a better random no. gen. changes rate of learning much.

.32 (.11) → Kernelizing the ends of $\vec{L}$: Bradw only used ½ of the kernel (?) – but any way, we can reflect the ends of $\vec{L}$ to continue them to get the kernel xfm!
i.e. $L_{T-2} L_{T-1} L_T \| L_{T-1}, L_{T-2}, L_{T-3} \ldots$ (continued as)   (and $\| T_1\ T_2\ T_3$ is continued as $T_4\ T_3\ T_2 | T_1\ T_2\ T_3 \ldots$
So no "kernel problems".

4.14.04

4TM

spec

70 : 115.90 : Another viewpoint to forget! That we use the "same GPD" for the present problem as for the general problem of Improving the induction Algm. In an early version of 1 IDSIA reports (probably still Abstract) I had this GPD that did both regular problems & "Higher order problems" like devising a better algm. to solve regular problems. I spoke of a "GPD" to do this — Running of a "GPD" in an abstract way: But if we consider a PD to be a program that effectively obtains P.D.'s;

56 Then its a more practical, usable way to look at the problem. → (.20)

Reading my most recent IDSIA Report, is suggestive: Even the 1975 TBilisi Paper. The idea is that the lowest level, "mainline" problems of TM should be similar to the meta problem of improving the algm that solves the low level problem. So each soln of a "l.l. (low level) problem can go, (to some extent) into the "corpus" (or the solved corpus) of the h.l. (high level, meta problem) of improving the method to solve low level problems.

O The "L" system suggests one way to do this.

One thing I hadn't thought about enuf, is improvement by Logical Analysis.

The way I expected to do it was to first teach TM how to solve formal logical problems then teach it the correspondence betw other types of real problems, & the formalisms of deductive logic — so it can apply deductive logic to these new probs.

An possible approach would be to teach deduction as a kind of induction — otherwise nothing special about it, & its always possible that it will be "wrong".

O (P6) (115.8) This idea of a PD having a cc param seems very simple, & the idea of "improving a pd. by adding more cc" of 114.10 ff is one way of (partially?) ordering such d.f.s. The defn. of the Univl. d.f. as the limit of a seq. of P.D.'s obtained from "FOR's" of successively larger CB, — is another way of doing this.

So we can have a vast panoply of defnt PD types, w different input forms different output forms & different relationships of cc to output pc. We probly need algms to translate from one PD type of I/O, to another.

I could have the Maintained L d.f. w. G values as a "Utility" & have TM devise different ways to use this utility for problem solving. Its something the system has to contend;

O Like a bunch of floating pt. insts. that we give TM to use as it likes, or some analog devices, or a Dictionary or Encyc.

Anyway, I've considered 2 extreme paths to TM! :

I) Phase 1, Phase 1½, Phase 2 Grammar of PSM's (← heurists, Grammar...)

II) GPD having pd used to guide search for "low level probs"; Same PD is used to Change/Algms (improve) used in [Probsolving of all kinds. (l.l.)

While II) is probly better in a theoretical sense, It is probly best to do I) because I get faster F.B., E.g. Its better to build successively better objects than to try to make the final object at once.

4.12.04
4 TM

10:11440 : How can we ↓ PCy ~~[crossed out]~~ cc? Well one way is to perhaps drop the idea of a single/general purpose problem, & use a TSQ to educate the TM so it can have a better Guiding pd for LSch.

At first (as in Phase 1) we use a simple induction system to summarize ~~[crossed out]~~ ways found in TSQ in the past. There is probly a limited amount of cleverness in this Guiding P.d., but the abss. used by the system mite "automaticly" be configured so that the Guiding P.d. is as effective as possl. . . . remember the basic lng. being used is Universal, & — so it can "model" "simulate" any conceivable instruction set.

I've noted that the Lisp lang seems more suitable for PPM guiding the PD than Forth. — If this is indeed so, can Forth simulate Lisp in a way that makes PPM a "good Guider" for the resultant lang.? Well Forth can simulate Lisp by defining a set of Lispish instructions. One Q is — What kind of TSQ would bring this about? Essentially, we want Forth to define words → concat of those words tend to be useful ones & perhaps be worthy of definition. This definition of new tokens really modifys PPM a lot!

0

53 — Also in AZ141, which is a Lispish lang, I also define new tokens. → 116.20

~~[crossed out]~~ [SN] I am concerned that in many induction environments, I need a fast approx. pc values for the "Guiding PD". One area of induction where speed has been emphasized is Speech recognition; Analog input — digital output.

[SN] In a TM system as contemplated I need:

0

1) Alg. to generate new cands
2) Fast as possl. way to test cands (Gorc)
3) Way to remember params of each cand, & its Gorc.
4) method of mapping 3) into 1)

Possl. use of GP to discover good Algm. for Cand. Generation.

In GP/GA we do have to store/remember population. Here in the "L" w/ PPM method we don't have to, but the L distribn (plus Gorc info) has all info in population, but in a more useful form. So while it does ~~[crossed out]~~ probly require more memory than GP, it doesn't require much more ~~memory~~ memory (That mite be easier to do size sweeping in GA/GP population??) — or more diff(x)!

0 It would be nice if we could do a hy computing capacity run of GP to get a good cand generating Algm. — Or a good pd generator to generate cands.

32 If this were possl., it mite be best to use 1/10 the IPC (Info proc cap) a system ~~[crossed out]~~ & use its output for the next run of cand generators, ect. It is probly much more efficient to generate improvements in the system sequentially ("Bootstrapping") than doing it all in one big run.

The Gorc must (perhaps) use a diffrnt set of problems from the problem of Generating good models.

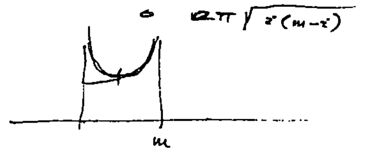
36 → I have to work this out much more carefully!

HA! ⓒ What .30 Amounts to (if one uses ~~[crossed out]~~ 1/2 instead of 1/10 in 32, is updating cand generation algm. during the Lsrch! — I.e. learning while searching.

Still (.36) is true.

4.10.04 /14
4TM

.20 : So $\sum_{0}^{m} \frac{1}{2\pi\sqrt{z(m-z)}}$ $= \frac{1}{2\pi}\sum_{0}^{m} \frac{1}{\sqrt{z(m-z)}} \approx$



$\int_0^m \frac{dz}{\sqrt{z(m-z)}} \approx \sum_0^m \frac{dz}{\sqrt{z(m-z)}}$ — 2π except singularities @ z = 0, m.

so this is $\approx \frac{1}{2}$

.05 So 113.87 becomes $\frac{e^m}{2}$. Now I have to go back to 113.12 (or maybe 113.11!) to see just what it means!

Conclusion:

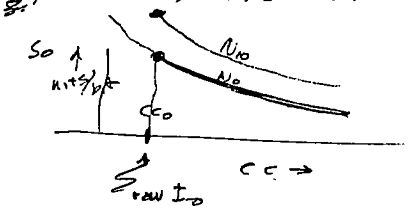
$$\sum_{z=0}^{m} z^z (m-z)^{m-z} \cdot \frac{m!}{z!\,(m-z)!} \approx \frac{e^m m!}{2} \approx \frac{m^m}{2\sqrt{2\pi m}}$$

I Guess t. original (error) difficulty started w. 109.06

SN A measure of Utility or Merit, of an induction System: Suppose an induction system I_0 used in a direct manner, is able to get an entropy of N_0 bits/symbol of a particular corpus.

By using I_0 to direct (≈ Lsrch) we can decrease this to N_1 bits/symbol, but it would take more cc.

By ↑ t. cc, we can obtain further ↓ of bits/bd.



bits/bit = $N(I_0, cc)$. (denoted I_0)

Now it may be that t. form of t. curve $N(I_0, cc)$ is "universal" i.e. It's t. same for all Induction systems! I.e. we just have to know one pt. on t. $N(\ ,cc)$ curve & we can immediately obtain all of t. curve.

So if I_{10}, say had a min N value of N_{10} w. a minimum cc of cc_{10}, then we just get

.20 $N_{10} > cc_{10}$ on t. curve $\vdots$ $N_{10}(I_{10}, cc_{10}) \triangleq N_{10}$; then

$N_{10}(I_{10}, cc) = N_0(I_0, cc) \cdot \frac{N_{10}(I_{10}, c_{10})}{N_0(I_0, c_{10})}$.

The figure of Merit of a Ind system in t. bits/symbol at some fixed, standardized cc level.

T. way we are able to get better n/symb. from an induction system I_0 —

we use I_0 to construct t. guiding PD, for an Lsrch to find a Ind. systems

27 { w. better n/symb. Of course t. choice of models, etc. for t. Lsrch is critical,
28 & is important in determining how entropy ↓ w. cc.

. But given .27–.28, and initial Ind system I_0, we can get a new ind system I_1 w. a larger cc.

.30 Presumably, we can repeat this operation indefinitely, getting $I_0, I_1, I_2 \ldots$ each w. more cc.

Would we get as good, or better results by just using "one stage of simplify" & a very large c.B. for Lsrch? (Perhaps not! t. successive I_0, I_1, I_2 gets better & better models, I, "Just using bigger CB"/Method uses same cong every time. Th. Successively better I_i is more like Phases, → Phases parts → $\forall$ sets — some ~PSG: Discovery of words.

On t. other hand, using a previously discovered PD to guide a new Lsrch & a Big ↑ in cc! —

So for this "Much better PD" one pays Heavily.

Speaking loosely, we know that using a ≈ null induction system to guide Lsrch, we can approach t. μ that generated t. corpus, arbitrarily closely — — — but w. much expenditure of cc.

4-10-04
4PM

(12:40)
1021112:40 :

We use precorpus code $\Delta abcd$ — the alphabet. $aa\Delta$ "a" is the seq being defined to be "α"

1. first Δ means "end of a data is something or one". The second Δ means "end of precorpus".

We assume the alphabet is known — but the first repeated symbol must be the start of data.

What then "corpus" means, is that symbol "α" now follows "abcd" effectively

$\Delta abcd\alpha aa\Delta \ldots$ Then the regular corpus follows.

Coding the $aa\Delta$ of the corpus costs for a (abcdΔ ...) at that pt. $\Delta\Delta$ would mean "end of precorpus".

so pc of a is $\frac{1}{5}$; next symbol could be abcdΔ so pc of Δ is $\frac{1}{6}$, next Δ could be abcdΔ, so its pc is $\frac{2}{7}$.

(the "regular corpus")

The cost of the corpus: N symbols, "a" occurs m times.

e^{-x^2}, e^{-Ax}

A naive code gives the symbol a a pc of $\frac{m}{5+N}$ — 5 is # symbols in corpus

A less naive pc would be given by the "factorial formula".

When we do i α's then we do $m-i$ a's: so the pc of a version i.

$\sum \frac{e^m}{2}$ see 119.00-05

so we have something like $\sum p_a^{\,i}\, p_\alpha^{\,m-i} \cdot \frac{m!}{i!\,(m-i)!}$ and $p_a \propto i$, $p_\alpha \propto m-i$

$\rightarrow$ So $\sum \approx i^i\, (m-i)^{m-i} \cdot \frac{1}{i!}\frac{1}{(m-i)!} \approx \frac{e^i \cdot e^{m-i}}{2\pi\sqrt{i\cdot(m-i)}}$ So $\sum_{i=0}^{m} \frac{e^m}{2\pi\sqrt{i(m-i)}}$ except for $i=0$ & m!

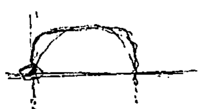
The $i=0$ & $i=m$ terms have to be treated separately — I think they are.

So, it looks like I can find the variation of P_{Tot} as i varies!

The $i=0$; $i=m$ terms; $0^0 m^m \cdot \frac{1}{0!\,m!} \approx \frac{e^m}{\sqrt{2\pi m}}$ so the 2 end terms are $\frac{e^m}{\sqrt{2\pi m}}$

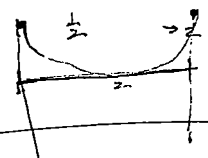
$$So\ \sum = e^m\left(\frac{2}{\sqrt{2\pi m}} + \sum_{i=1}^{m-1}\frac{1}{2\pi\sqrt{i(m-i)}}\right)$$

& the other terms are $\frac{e^m}{2\pi\sqrt{(m-i)\,i}}$

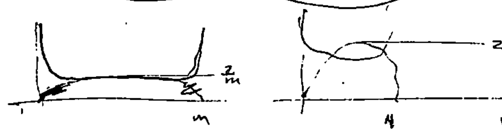


$\int_{\epsilon}^{1-\epsilon}\frac{1}{\sqrt{x(1-x)}}$ $\int_0^m \frac{dx}{\sqrt{x(m-x)}}$ $= \int_0 \frac{dx}{\sqrt{x(m-x)}}$ $=\pi$ substitution $x=\frac{y}{m}$

$\frac{1}{2}^2 - x^2$



$\frac{m}{2}$ $S > 2.$



$\sqrt{x(1-x)} = \sqrt{(y+\frac{1}{2})(y-\frac{1}{2})} = \sqrt{y^2-\frac{1}{2}^2}$

$x = y+\frac{1}{2}$ $\sqrt{y+\frac{1}{2}\cdot y-\frac{1}{2}} = \frac{1}{\sqrt{y^2-\frac{1}{2}^2}}$ $= y+\frac{1}{2}$ $1-(y+\frac{1}{2}) = \frac{1}{2}-y$

No!

$$\int_0^a \frac{dx}{\sqrt{x^2-a^2}} = \ln\left(x+\sqrt{x^2-a^2}\right)\Big|_0^a = \cosh^{-1}\frac{x}{a}\Big|_0^a = \cosh^{-1}1 - \cosh^{-1}0$$

$h\cos^{-1}1 = 0$; $h\cos 0 = 1$ Yes.
$h\cos^{-1}0 = \infty$? $h\cos 100000$
$h\cos 100 = 1.3\times10^{43}$, $\int(a^2-x^2)$

$$\rightarrow \int_0^a \frac{dx}{\sqrt{x^2-a^2}} = \ln\left(x+\sqrt{x^2-a^2}\right)\Big|_0^a = \ln a - \ldots\ \text{complex!}$$

$\frac{1}{dx}\ln\left(x+\sqrt{x^2-a^2}\right)$ $\frac{1}{x+\sqrt{x^2-a^2}}$

for $x=a$, this is real, but for $x=0$ it's complex! $\ln(\pm i\cdot a)$

$$2\cdot\int_0^{\frac{1}{2}}\frac{dx}{\sqrt{a^2-x^2}} = \sin^{-1}\frac{x}{a}\ \text{or}\ -\cos^{-1}\frac{x}{a}$$
$\frac{1}{2}$ $= \sin^{-1}\frac{x}{\frac{1}{2}}\Big|_0^{\frac{1}{2}}$ $\sin^{-1}1 = \frac{\pi}{2}$, $\sin^{-1}0 = 0$

So $\int_{-\frac{1}{2}}^{+\frac{1}{2}}\frac{1}{\sqrt{\frac{1}{2}^2-x^2}} = \pi$

4-8-04
4 TM

$$Z \equiv \frac{m!}{\sqrt{2\pi}}\beta^{-m}\sum_{j=0}^{m-1}\beta^{j}\frac{1}{j!\sqrt{m-j}} = \left(\frac{m}{e\beta}\right)^{m}\sum_{j=0}^{m-1}\beta^{j}\frac{1}{j!\sqrt{m-j}}$$

$Z = Z(m,\beta) =$

m=2		20	
1.3		705	← $\beta = \frac{m}{2}$
1.022		2.9	← $\beta = .8m$
.6	m=6	1.8	← $\beta = .9m$
.52	m=10	1.28	← $\beta = m$

$\beta = N\rho_{T\alpha}$ so $\rho_{T\alpha} = \frac{\beta}{N}$

$\delta = \frac{m}{N-km} \approx \frac{m}{N}$

$m > \beta$ in all cases of interest.

1) Check the math, 2) Do eqns in Maple in Σ form. 3) Do eqns exactly, w.o. approx — see if it makes any difference — also see if it gives more "reasonable" results!

Seems that $\delta = \frac{m}{N-km}$ maybe strictly wrong: should be $\frac{m}{N-km+m} = \frac{m}{N-(k-1)m}$

$\delta = \frac{m}{N-km+m}$ $\quad 1-\delta = \frac{N-km}{N-km+m}$ $\quad (1-\delta)^{N-km}$

$(1-\delta)^{A}$

$\ln(1-\delta) \approx -\delta + \frac{\delta^2}{2}$

$e^{\left(-\delta+\frac{\delta^2}{2}\right)(N-km)}$

$e^{-\frac{m\cdot(N-km)}{N-km+m}}$

$e^{-\frac{m}{1+\frac{m}{N-km}}}$

"Brushing a bit under the rug" say .25 is ok.

$\left(\frac{m}{N\,P_-\,e}\right)^{m} P_- \ldots$ $\quad NP_- \equiv \beta$ so $\left(\frac{m}{\beta e}\right)^{m} P_-$

Should we sum m' from ϕ to m? I guess it's a logical code: it consists of making the decl. of α, but not using it at all.

$$\rho_-\left(\left(\sum_{i=\phi}^{m}\left(\frac{i}{\beta e}\right)^{i}\frac{m!}{i!\,(m-i)!}\right)+\rho\right)$$

$\left(\frac{i}{e}\right)^{i} \approx \frac{i!}{\sqrt{2\pi i}}$ if $i \neq 0$.

$$\rho_-\left(1+\sum_{i=\phi}^{m}\beta^{-i}\left(\frac{i}{e}\right)^{i}\frac{1}{i!}\frac{m!}{(m-i)!}\right)$$

So do sum from 1 to m, but add in term. $i=0$ term. $\lim_{x\to 0} x^{x} = 1$ $\quad x\ln x \to 0$ as $x \to 0$. which is 1.

$$\rho_-\left(1+\sum_{i=1}^{m}\beta^{-i}\frac{m!}{\sqrt{2\pi i}\,(m-i)!}\right)$$

$$1+m!\sum_{i=1}^{m}\left(\beta^{-i}\frac{1}{\sqrt{2\pi i}\,(m-i)!}\right)$$

← Peak of summand occurs ≈ when $\beta = m-i$ for each $i \to i+1$, since summand ↑ by factor of $(m-i)$ & ↓ by a factor of β. (It's also ↓ by factor $\sqrt{\left(\frac{i+1}{i}\right)}$) This would be a poor size is not affected when mult by e^{-Ax}

Easier to compute $i!$

$j \equiv m-i$
$i = m-j$
$-i = j-m$

if $i=1$, $m-j=1$; $j=m-1$
$i=m \to j=0$

$$1+\frac{m!}{\sqrt{2\pi}}\sum_{j=0}^{m-1}\beta^{j-m}\frac{1}{\sqrt{m-j}\,j!}$$

$$Z(m,\beta) = 1+\frac{m!}{\beta^{m}\sqrt{2\pi}}\sum_{j=0}^{m-1}\frac{\beta^{j}}{j!}\frac{1}{\sqrt{m-j}}$$

! Is it at all reasonable that $Z(m,\beta)$ should be > growth if $m \geq \beta$; or — for all m?
$Z(m,m)$ slowly ↑ w. m.

(Could this slow ↑ of $Z(m,m)$ w. m be an approx error?
I would think that $Z(m,m)$ should be $\equiv 1$ for all m, because it makes extra codes of identically same pc's, so the $(1-\delta)^{N-\ldots}$ factor, which is < 0; must add up to 1 viz $\sum \frac{m!}{m'!(m-m')!}$

This factor is exactly $(1-\delta)^{N-km}$ so is $\sum_{i=0}^{m}(1-\delta)^{N-km+i}\frac{m!}{i!\,(m-i)!} \equiv 1$?

Seems like the $(1-\delta)^{N}$ factor should exactly cancel the Σ

Try is using exact summations and exact δ values, etc.

Try coding the following! Been say: we define 1 sol. (one of the symbols: Substitution of the decl. code α in many ways should give $\equiv \phi$ change in pc (I think). If not then I'm losing some pc somewhere...

(113.00 sec)

4TM

0 ! (Spec 109.40) ! T. Stirling approx is very good for m > 1, say, "2!" $\frac{2^2}{e^2}\sqrt{2\pi \cdot 2} = 1.919$

But 0! does not work w. Stirling ! "1!" $\frac{1}{e}(\sqrt{2\pi}) = .922$ not bad!

In 109.25 ! $P_{T\alpha} \sum_{m'=1}^{m} \left(\frac{m'}{N P_{T\alpha} \cdot e}\right)^{m'} \frac{m!}{m'!(m-m')!} = m! P_{T\alpha} \sum_{m'=1}^{m} \left(\frac{1}{N P_{T\alpha}}\right)^{m'} \cdot \left(\frac{m'}{e}\right)^{m'} \cdot \frac{1}{m'!} \cdot \frac{1}{(m-m')!}$

Since there are no neg terms, i m' = 1 to m (never zero) $\left(\frac{m'}{e}\right)^{m'} \frac{1}{m'!} \approx \frac{1}{\sqrt{m' \cdot 2\pi}}$ | $\left(\frac{i}{e}\right)^{i} \cdot \sqrt{2\pi i} \approx i!$

.08 so $\frac{m! P_{T\alpha}}{\sqrt{2\pi}} \left[\sum_{i=1}^{m} \left(\frac{1}{N P_{T\alpha}}\right)^{i} \frac{1}{(m-i)! \cdot \sqrt{i}} \right]$ $z \approx \sum_{i=1}^{m} \beta^{-i} \frac{1}{(m-i)! \sqrt{i}}$

$\frac{m^{m+\frac{1}{2}}}{e^m}$ $N P_{T\alpha} = \beta$, Z; $N P_{T\alpha} \approx < m$

j = m−i i = m−j; −i = j−m

10 $\frac{Z}{m!} \approx \beta^{-m} \sum_{j=m-1}^{j=0} \beta^{j} \frac{1}{j! \sqrt{m-j}} = \beta^{-m} \sum_{0}^{m-1} \beta^{j} \frac{1}{j! \sqrt{m-j}}$

Study $Z(m,\beta)$, $Z(m,m) \to m^{-m} \sum_{0}^{m-1} m^{j} \frac{1}{j! \sqrt{m-j}}$ as a funct of m.

m = 1 perhaps doesn't occur ~~[illegible]~~ Z(1,1) = 1

$Z(2,2) = 2^{-2} \sum_{j=0}^{1} 2^{j} \frac{1}{j! \sqrt{m-j}} = 2^{-2}\left(\frac{1}{\sqrt{2}} + 2\right) \approx .675$

Z(3,3) = . M^(M+½)*exp(−M+1/12/M)*2.506628 ($\sqrt{2\pi}$) 2.506628

To evaluate Z(m,β) (.10)

$A_j = \frac{\beta^j}{j!}$: $A_j = \frac{\beta}{j} \cdot A_{j-1}$.

```
A=1 : S = m^(−.5)   "B is for j=0
For j = 1 to m−1
  A = A*B/j : S=S+A*(m−j)^(−.5)
Next
Print M, B, S·B^(−m).
```

B35! 4TM 111.BAS

$1! \approx \frac{1^1}{e^1}\sqrt{2\pi} = .92213$ × 1 1/12 = .99898

$2! = \frac{4\sqrt{2 \cdot 2\pi}}{e^2} = \frac{8\sqrt{\pi}}{e^2} = 1.919004$ × $(1 + \frac{1}{24} =)$ 1.99896

M can be quite large: k is a one that was << m pos

Even w. β = ½M, and large Ms we get very small Z for ~~[illegible]~~ large M !

woops! I left out the m! factor. $m^m \times \exp(-m)$ × (2×3.14159×M)^(.5)

~~putting~~ putting β = m/2 : for m = 2 to 20 Z = 3.2 to 177.

~~[illegible]~~ I have to mult by $P_{T\alpha}$ ÷ $\sqrt{2\pi}$ (≈ 2.5...)

$P_{T\alpha}$ can be quite small if m is large. If is + original cost of doing α ~~[illegible]~~.

If may be that they is not necy! But one can do ~~[illegible]~~ α a lot cheaper!

divided by $\sqrt{2\pi}$ we get 1.3 to 70.5 (m = 2, m = 20)

0 for β = .8 m ~~[illegible]~~ $\frac{Z}{P_{T\alpha}}$ goes from (.022 to 2.9) (m = 2, m = 20)

		m = 2	m = 6	m = 20
for β = .9 m	$\frac{Z}{P_{T\alpha}}$( ) =	.6	1	1.8
			M = 10	m = 20
for β = m	" =	.52	1	1.28

It would seem that for β = m, ~~[illegible]~~ the der. of α should not help for any value of M !

Also, it would seem that for any ~~[illegible]~~ $\frac{\beta}{M} = c$ & c < 1 then

for large enuf M, Z should be arbly large.

$\beta \equiv N \cdot P_{T\alpha}$ ∴ $P_{T\alpha} = \frac{\beta}{N} < \frac{m}{N}$

4.8.04
4 PM

→0 : also: look at $m = 2, 3, 4$ etc (common cases) — w. large k, hvr. k will be as m+.

Re: Much cc for colns: For PPM they only use $k \le 5$, say w. v.g. results — even in PPM* w. large k allowed, many of t. large ones were very infrequent.
So perhaps my original factorial version of $P_{T\alpha}$ — "exact"
If its easier to calculate using "mixed" version, using a $(1-\delta)^m$ factor to compensate for t. defn. of α.

While 108.40 mite be o.k. for obtaining $P_{T\alpha}$, it seems very expensive. On t. other hand, using t. case counts of t. tokens as in t. factorial method of Z141 would seem to be catastrophically poorer (because of commonly occurring "correct predns") than 108.40! So Z141 may be better

09 than LZ, (if even that!) but probly much worse than PPM! (.13)

→ T. expense of 108.40 isn't altogether outrageous! We will be keeping track of t. cumulative pc's of every point in t. corpus. This mite be done using a kind of "floating pt.", in which I use fixed pt. multiplication, but I keep track of shifts needed to keep "mantissa" ≈ 1.

13 :09 One could "use" Z141, but use 108.40 for evaluating $P_{T\alpha}$. Perhaps t. system would start out pretty much like old Z141, but gradually improve as t. predictions improved as $P_{T\alpha}$ was computed w. more accuracy. This is perhaps t. best way to do "Z141" anyway!
We may start out using t. "factorial formula"

0 : I do need to get that ν Z141 coding straightened out! I will probly need it when I do Grammar-discovery to augment "Phase 1". I think I'd need it for (.13) even! Perhaps by proofreading Sol 64!

(4.14.04) I can "sort of" see why I felt that OOPS' language wasn't as good as Lisp-like AZ141 —
In AZ141, t. sub-sequences always (or usually) have meanings as sub-functions.
In OOPS's forthish lang, this does not seem to be true. One could, perhaps, in OOPS adopt a very forthish programming style, in which almost all pgms were short seqs of dictionary entries — in which case, t. desired "context structure" utilized by PPM mite work o.k. In fact PPM mite impose a "style of pgmmg" on t. system in a way so that PPM would be useful (a sort of self-confirming hypothesis enforcer ©) — in which case

30 one could use (perhaps) almost any universal formalism that could express t. desired solns as "short codes". Hvr, w. some universal langs, it may take a much larger corpus before t. system begins having usable PPM-type regularities. → On t. other hand this distortion mite be implemented by a suitable initial TSQ.

[SN] "To first order" © PPM does everything that Lsearch do, but cheaper —
Hvr. on "second order" (advanced lang) PPM may be unable to do t. equiv. of defining Nparsts + obtaining t. implied "Grammar".

00:108.40 : If α is assigned pc of δ, all tokens will have their pc multby $(1-\delta)$. [Token]
α contains k tokens. α occurs m times in (augmented by α) corpus.
The geom mean of the pc's of the ~~symbols~~ tokens of α in the corpus is $\boxed{P_{T\alpha}}$
Therefore, before intro of α & include the final "α"; N tokens in the corpus,
factor of change of pc of corpus brought about by use of α, m times is:

.06 $$\approx \left(\frac{\delta}{P_{T\alpha}}\right)^m \cdot P_{T\alpha} \quad (1-\delta)^{N-km}$$ [≈ pc of definition of α.]

$$(1-\delta)^{\frac{\delta(N-km)}{\delta}} \approx e^{-\delta(N-km)}.$$

0 $$\approx \left(\frac{\delta}{P_{T\alpha}}\right)^m \cdot P_{T\alpha} \frac{e^{\delta km}}{e^{\delta N}} = \left(\frac{\delta \cdot e^{\delta k}}{P_{T\alpha}}\right)^m \frac{P_{T\alpha}}{e^{\delta N}}$$

$\delta^A \cdot (1-\delta)^B = \max$
$\frac{\partial}{\partial\delta}(A \ln\delta + B \ln(1-\delta))$
$\frac{A}{\delta} = \frac{B}{1-\delta}$
$B\delta = A - \delta A$
$\delta = \frac{A}{A+B}$

So perhaps assign value to δ so that this total pc is max

The values dependent on δ is $\frac{\delta^m}{e^{(N-km)\delta}} = \frac{\delta^m}{A^\delta}$ $\quad A = e^{(N-km)} \gg 1$

$m \ln\delta - \delta \ln A = \max \quad \frac{m}{\ln A} \cdot \ln\delta - \delta = \max$

$\frac{m}{\ln A}\frac{1}{\delta} - 1 = 0 \quad \delta = \frac{m}{\ln A} \quad \ln A = N-km \quad : \quad \boxed{\delta = \frac{m}{N-km}}$

$\frac{m}{\ln A} = \delta$

$e^{-\delta(N-km)} = e^{-\delta(N-km)} = e^{-m}$

So .06 $$= \left(\frac{m}{(N-km)P_{T\alpha}}\right)^m e^{-m} = \left(\frac{m}{e(N-km)\,P_{T\alpha}}\right)^m$$ So we get the factor of e again! [But it ≈ cancels out! !!]

0 $$= \left(\frac{\delta}{e\,P_{T\alpha}}\right)^m \cdot P_{T\alpha}$$ [→ correction factor] $\delta = \frac{m}{N-km}$ $\quad = \frac{1}{\frac{N}{m}-k}$ | I assume $km \ll N$ so $\delta \approx \frac{m}{N}$.

.23 $$\approx \left(\frac{m}{N P_{T\alpha} \cdot e}\right)^m P_{T\alpha} \quad \left(\frac{m}{e}\right)^m \approx \frac{m!}{\sqrt{2\pi m}}$$?

$B_{T\alpha} = \frac{\beta}{N} \lessapprox \frac{m}{N}$

Next, we have to consider all poss codings [Does m mean m'?] 2^m of them.

.25 $$P_{T\alpha} \sum_{m'=1}^{m} \left(\frac{m'}{N P_{T\alpha} e}\right)^{m'} \cdot \frac{m!}{m'!(m-m')!}$$

[The m=1 may not be a good deduction by itself, it but a lot ≥ anyway the Math easier]

$m'! \approx \frac{m'^{m'}}{e^{m'}} \cdot \sqrt{2\pi m'}$.

We expect β to be $< m$

$\beta \equiv N P_{T\alpha} \approx m$ to petition of how big β is. | i.e. α occurs more freqly than $P_{T\alpha} \cdot N$

$$\approx P_{T\alpha} \sum_{m'=1}^{m} (e N P_{T\alpha})^{-m'} \frac{e^{m'}}{\sqrt{2\pi m'}} \frac{m!}{(m-m')!}$$

gets rid of e factor!!

0 $$= \frac{P_{T\alpha}\, m!}{\sqrt{2\pi}} \sum_{m'=1}^{m} (N P_{T\alpha})^{-m'} \frac{1}{\sqrt{m'}} \frac{1}{(m-m')!}$$

$j = m-i$
$m - j = i$

thus $$\sum_{j=1}^{m} \beta^{-j} \frac{1}{\sqrt{j}} \frac{1}{(m-j)!} = \sum_{i=m-1}^{0} \beta^{i-m} \frac{1}{\sqrt{m-i}} \frac{1}{i!}$$

= some function ~~of m & β.~~ $$= \frac{1}{\beta^m} \sum_{0}^{m-1} \frac{\beta^i}{i!} \cdot \frac{1}{\sqrt{m-i}}.$$

I could plot it for various β, m. — But β will be $< m$.

So do eval. for various large m, & $\beta = .9$, m, .8 m, etc.

Go over those Eqns carefully before going! Getting rid of the $\frac{1}{e}$ factor is a great relief!

Do it more exactly w. $(1-\delta)^{\cdots}$ using second order formulas
~~.06~~ .23 → .25 ... use the ... e^{x-x} for x!

→ (SPSC 101.00)

00:107.40: Q: Is t. kernel width ∝ t? Insertion into L at various pts, will occur w. different T values. Hvr, at each T value, t. total length of L is ≡ t. The width of kernel will always be a certain fraction of t. length of L (≡ t. length of L ≡ t). If we have a fixed L (or a L of fixed length — obtained by deleting a card whenever one inserts one), Then we have a kernel of fixed width.

[SN] Adding a new token is certainly cup-t at t. beginning — yet it does seem to screw up the needed shape of t. kernel! Another disturbance is Later when a new token is defined (as in OOPS). In t. first case (early [illegible]) we have to give equal pc to several possible symbols — Can't be done w. a "procorpus", because symbols in procorpus are at different distances.....(?) no! — those are distances in corpus space (T direction). We are concerned w. "L" space ... "L" direction.

0 For t. design of sequences of instructions; we can have t. "procorpus" in t. form of K different cards: ΔT_1 (start) ; ΔT_2 ; ... ΔT_K. T_i is ith token. (or (): $\wedge T_1, \wedge T_2 \ldots$ ($\wedge$ = null context)

We could probably add in definitions as well, in this way: a bit [illegible], hvr.

Error of t. kernel is pretty second order effect, but t. relevant error is in t. log of t. kernel since we are interested in maximizing t. pc of t. soln — t. product of pc's of tokens $= \exp(\Sigma \ln(\text{tokens}))$.

T. idea of this "kernel" has one (apparent flaw). T. lexical ordering of t. tokens influences

10 — pc a lot. When one sorts t. corpus Lexically → L, this automatically occurs. (.31) is another way to deal w. it.

One way to avoid this — The only measure of similarity of 2 contexts is t. length of their common suffix. This seems to be used in conventional PPM & PPM*. The relative wts of these common suffixes is (perhaps) shared by all suffixes. An alternative way would be to view each suffix as a pem — a few seq. w. a procorpus (of wt. ½, perhaps). So t. past success of t. suffix is

.25 used to help get its wt. (.33) } Is this ≡ 2141?

Q: To what extent is the "escape" probability close to that obtained by using a "procorpus"? In both cases t. pc of t. new token is a [illegible] $\frac{1}{T}$ (T being no. of tokens thus far) (or is $\propto \frac{1}{T}$)

(Apparently) t. only way to do this, is to define K [illegible] (There are K different Tokens) I. [illegible]

So t. [illegible] — 1 for each context, & consider t. set of ways to code t. corpus using that definition

30

35 (.20) In t. PPM discussion of ~1995 (Witten, Cleary, Moffat) they use a "Trie" structure that somehow seems to avoid the error of Lex ordering — but it may be more Expensive in time and Memory.

33: (.25) Expanding this a bit! To code using α & [illegible] This is a particular suffix ending in "a". t. suffix to a is of length k.

First define α in t. procorpus. pc of definition will be ≈ product of (pc's in t. corpus of its tokens).

Also, [illegible] pc of symbol denoting "definition".

In theory, we can compute the pc of each sequence of tokens replaced by each α.

40 This is done by looking at t. cumulative pc of t. corpus at t. beginning and ends of each α insertion

APM

0:106.40: Hvr, I realy have no good idea as to how long it takes to generate a best-a-case. Currently, it takes $\approx 2000 \cdot \ln D$ clocks to put 1 token into $\vec{D}$. Also, 1 (or 10) G by of Ram may be enuf for all that I need to do.

In T≡2T (which seems simplest to pgm) + Bottleneck seems to be t. compa. eff. pc'al tokens.

Try to see how much accuracy I need in various token pc's.

As is, I don't have a clear idea as to how much Memy I need, how fast, how good a TSQ is needed to get to phase 2.

For Token pc calcus: I may be able to simulate t. kernel by a Laplace xfm, Rn convolute it using $x_n = \alpha y_{n} + (1-\alpha) x_{n-1}$ — using many α's w/ + a possible negativeuts. This could be linear in SSZ. As is, I have no idea as to what t. kernel looks like.

I could just accumulate an empirical kernel, as 103.05 : Hvr I had idea that it was a vector kernel!

One kernel for each of t. Token types!

("Thinking out loud"): When one asks for t. pc of a context: we look at longest (interval matches) in corpus. — How far along "L" do we have to go before we find a particular Token? t. L sequence of tokens can be treated as a sequence of "Token vectors"; (A token vector is a unit basis vector in Token space ≡ k dim space, k dif. tokens)

So we convolute $\vec{L}$ w. t. (scalar kernel) + we get a sequence of (smoothed) vectors.

To store these smoothed vectors would require $k \cdot N$ bytes or words. (N = no. of tokens in corpus).

Hvr, because of t. smoothness of t. kernel, it shouldn't be nec. to store so much info (something like only 2f points per second for a "f" bandwidth signal). If t. kernel is of width W, it may be nec. to store every Wth point of a suitably filtered form of $\vec{L}$: perhaps a "boxcar" filter of width W or $W/2$ would be enuf.

---

1) Say kernel is of width w. — Store every w or $\frac{w}{2}$th point of some filter of $\vec{L}$.

2) We don't need $\frac{N \cdot k}{w}$ points at all: Because of the $\frac{1}{N}$ compression as we go along, — the kernel "sort of" gets wider as we move along t. seq. It may be that t. no. of pts we need to store is indep of N(!). If t. kernel is $\approx \frac{1}{100}$ as wide as "N" then may be we need to store only 100 vector pts.!

So its impt to know just what fraction t. width of kernel is of t. corpus, N. — Will vary w. how accurate PPM is for that kind of corpus. Width will be $\propto$ bits/char compression perhaps.

3) Actually, t. entropy of t. corpus depends not so much on t. width of t. kernel, but on how rapidly t. "L" corpus shifts from emphasis on one Token, to emphasis on another.

.29 4) Consider t. follg. form for "L": (Token$_1$ m times)(Token$_2$ m times)(Token$_3$ m times)...(Token$_k$ m times).

If t. width of kernel is << m, so transitions will be short. T. correct Token prodn will usually be t. most likely one. Other times, it will be t. second most likely one. With only 2 symbols, t. entropy per symbol is < 1 bit/symbol (and actually << 1 bit/symbol). It would seem that BW's method (≡ Bzip2) would, in t. present case, do better than t. kernel, i.e. it would pick t. closest correct context — which would be correct, except at boundarys betw. symbols (i.e. transitions betw. symbols). BW would give very low pc's to tokens at transition, because they are not t. most recent, + haven't occured in a long time.

With a L of form .29, we can compute what t. kernel would be like! I think it would be very narrow: say 3 symbols wide (width apparently indep of position in corpus!!?) ≡ ?!

Another Q: Does a corpus exist ∋ it's L is describ. in .29?

4 TM

18:10β
−19

23 cHE ≈ 24k Books

01:106.40 : get this local concentration of cc. The rate of which this occurs will depend much on how our algm for f & pc's of contexts of hy G cands. This algm should depend on how near t. G of a cand is to t. best G's thus far — so as we climb t. hill, pc's cands not longer in t. "top percentiles" will actually ↓ — & t. pc's assoc w. their contexts will ↓.

O.K.: So write up all 3 Lrng Models in a little detail. Devise suitable notation — to desb. them & to facilitate ppms.

→ Q: Is there any scoring modifn. of $\vec{L}$ & its applicn, assoc. w. t. fact that t. ppms will be very short — <10 or 20 chars? "Δ" will occur often in t. "corpus" in L. I think there's a relevant recent discussion in deciding if I needed 2 symbols Δ & δ for start & stop.
↳ See 97.14 — 98.14 At 98.12-19, I that I had "Boundary Control"!

SN Interesting t. present system on problems others have worked on!. Consider that our system will work it has good, reusable contexts! These can be acquired by suitable initialization or by a suitable **Pre-TSQ**. It has to acquire functions, ect, that can easily be put together using ≈ PPM. **Presumably**, after it has solved a suitable large batch of problems, it will have to have such a set of functs, prefixes. The set of prefixes will be a function of t. initial set of primitives, and t. TSQ.
Perhaps it is possl. to Design a set of primitives ∋ PPM is adequate for almost all prob solns. (if one started w. a suitable "TSQ".

— i.e. in expressing functions: PPM is ≈ a "Bernoulli Grammar".
One Q is: to what extent is CFG form necy? We do use If ○ then ○ else ○ or just If α then β. T. boundaries of α & β have to be somehow indicated.
We have a function of α & β: This is easy to do in LISP, (I guess) — If α & β were ppms. Here, If in Basic & in longs. is t. for sequential control. If t. "IF" = False, we go on to next instr.
In Lisp; I guess If is a 3 argt. function — a switch 2 fencts.

How would it be done in a stack machine? Maybe put α & β, γ on t. stack (3 slots), then do t. If: we can end up w. either β or γ on TOS.

It may be clear by giving (Phase I) a large no. of probs to solve whose solns have cmt common features that PPM would learn enuf to be able to use those "common features":
On t. other hand, it may be necy to see in detail, how TM is supposed to solve various problems in t. TSQ.

When we find a soln to a problem, we should always continue t. srch to find a hyper pc soln (if possl).
— (Since w srch in [redacted] pc/cc order, it will have to be a soln of hyper cc as well.) T. reason hy pc is good, is that it will give good "concepts" for subsequent problems. Also simply having several solns to a problem is usually an economical way to get good / new concs & good wts. for concs.

So: In drawing up t. detaild. PEZT, 11, random models: Cost of Memory can be a contributor tot. cc. T. extent to which disc swapping can be used is unclear.~ See 101.09 for discussn, Numbers. 40 mby/sec write, 70 Mby/sec read. 25ns + 17ns = 42ns → ≈ 25 Mby/sec for read & write

4.04.04 ~~#05~~ 105

4TM ~~19:14 Ready~~ 20:13 "

0: ($ pc $04.40) It would seem that ~~It~~ Random Lsrch would not be v.g. in this respect, because a change in concentration of pc of part of cand drn would seem to not produce an immediate ~~[illegible]~~ effect in Random Lsrch.

|| Lsrch (& perhaps TS2T Lsrch) would seem to be more likely candidates: We would probly want to have a special srch as soon as a hy G cand is found. This might be possl. in Random srch as well!)

There is a particular variety of GA that does local srchs of this kind — Has a particular Name — (published in Italy?)

It would seem than normal || srch would be ~~be~~ not much better than Random srch in this regards since an ↑ in pc (of a particular region of cand drn space) merely ↑ the time share of that region — but this does not normally produce a "rush" to investigate ~~[illegible]~~ that area.

In TS2T Lsrch, if we try to do it in ≈ pc order, we could get this "localized investigation of ~~[illegible]~~ hy G pts".

On second thot, ↑ in G doesn't merely ↑ pc of a region — tho it ~~[illegible]~~ is able to change order of trials!

So: Present Prob: In view of 104.37; we'd like to be able to get ~~[illegible]~~ quick feed back from successful ~~[illegible]~~ Cands. (≈ local srch).

Would it be a good idea to write up a more detailed dern of each of 3 ~~of~~ Lsrch systems. (TS2T, ||, Random).

I would also like to refaint. ≈ 2D speedup of || & random over TS2T, if possld

At first Glance, it would seem that the normal methods of doing || & random Lsrch would be correct; that if a particular Cand had hy G & we wanted to give more wt. to it, we could do this by (slightly) increasing the pc's of each of its contexts (via the L structure). — So maybe we'd double or quadruple the pc of a cand & ~~[illegible]~~ ↓ by 2 or 4, the cc of solns (if any) "near" to that cand. ~~[illegible]~~ Hvr, this is a "business as usual" type of soln, & we'd want more "local srch" type of reaction. In Phase 2, I expected to use lrng during Lsrch since ~~[illegible]~~ Big thing — It may be that this "immediate FB" cannot be used ~~[illegible]~~ so effectively in Phase 1 .... That we have to got to Phase 2 to do it properly.

To put ~~more~~ more emphasis on ~~hy G cands~~ the "neighborhoods" of hy G cands, is a kind of "elitism" — Its values is .... Uncertain.....

One way to get this "local concentration" around hy G cands: If a hy G cand just has occured, in Lsrch, the "Proby" threshold of the system must be at a certain level; near that of the hy G cand. If we ↑ ~~[illegible]~~ the pc's of near by cands, as well as that of the hy G cand, we should place cc on them to bring them up to the new cc level — (This new cc level is cc the pc of the cand) which will prob cause some of them to finish & be evaluated; ~~[illegible]~~ If these cands have hy G, this will further ↑ ~~[illegible]~~ pc's of near by cands, ect. So we could → 106.00

: A possl. serious expense in Random Srch is Generation of random (Nos) trials. In most cases we don't need V.G. random nos., & so cheaper sources could be considered. Another possy is H.W. generation of random nos. At present, it isn't clear to me ① How many bits of randomness I need for each Token choice. ② what kinds of non-randomness are tolerable. — It may be that an exhaustive (non-random) seq. would be o.k. || See how much time it takes for PB35 a/o PBcc to generate random Nos. (subtract out time to do administration; by timing a ≈ "Null" loop.)

T. big Q is Time to do random srch v.s. time to do || deterministic srch.

Also, just how much time is needed for pc. calcn. in T≠2T Lsrch? ⊕ ? ?

My impression is that usually for ever random method, there is a non-random method that is about as good or better! — but not nec'ly easy to find!

O.K.! How to get pc's exactly; maybe not so expensively! At each point in "L" we store t. cumulative case counts of each of t. → TOKENS (since t. begining of L). After we have been running a long time, the $\frac{\partial}{\partial \ell}$ of these case counts will be about constant, & will be our "kernal." — So we average out a "kernal", which we update every once in a while. (Note: This is t. dem of t. ≈ constant kernal.)

Still, its not clear how we get Token pc's out of this!

T. density distribution ... $\left(\frac{\partial}{\partial \ell}\right)$ is over Tokens; it is quite diff't from t. "Kernal".

"ℓ" is motion along "L", say $\rho_\ell(\tau)$ is t. density at pt. ℓ of token, "rho" τ. K(x) is the kernal at ℓ (it is ≈ indep of ℓ). The pc of symbol τ is, then ∝ :

~~$\rho_\ell(\tau) K(x)$~~ $\int_{x=-\infty}^{x=+\infty} \rho_{\ell+x}(\tau)\, K(x)\, dx$.

Say $N_\ell(\tau)$ is t. total no. of times τ has been t. correct following symbol, up to point ℓ.

$\rho_\ell(\tau) = $ "$\frac{\partial}{\partial \ell}$" $N_\ell(\tau)$ T. quotes are because it is not a derivative but a "difference",

$= N_{\ell}(\tau) - N_{\ell-1}(\tau)$ is exact form, ... 13:12

So, we do know the function $\vec{\rho}_\ell$ (t. vector w. Token components) empirically as t. identity of t. tokens for each ℓ value. The convolution of t. kernal K(x) & t. density $\vec{\rho}_\ell$ can be done rapidly by Fast Fourier xfm. It takes time ∝ N ln N.

Couple of "Notes": ① T. convolution is essentially a "smearing process" & doesn't have to be done very accurately (for Cand. generation, at least)

② It really isn't exactly a convolution because t. kernal width varies a/c ∝ ℓ.

Re ②: I might distort the L distribution so that t. kernel was constant. May have to modify amplitudes well as dilatement of "ℓ" direction. → I think this will work. The dilated points need not be put at integral ℓ values.

Unfortly, this makes t. FFT impossl. (perhaps). I could move them to ≈ closest lattice pt. ← Yes,

— the kernal doesn't have to be so exact.

37 SN It would seem that immediate Feedback during Lsrch (would/could) be very imp! — i.e. one finds an apparently hy of direction & persues it — T. alternative is to have to wait until t. next T≠2T round (or equivt. delay).

If this F.B. usually is hy cc, then perhaps only do t. relevant "Update" when an unusually hy G card has been found. ≋ → (5 pot 105.00)

4.01.04   4 PM   # Random Lsrch .25   Note on PBCC: See "Variables and Scope" "Variables scope" for discussion of ... "share" "share equ...?" equiv.

103

20 :102.60 : A big Q, of course, is t. **Exact shape of t. Kernel**. I could get some ideas for this from looking at how PPM, PPM* = BUT it improve / macro ... of Peach work.

Try various forms & see how good they are at compression { NB. Each kernel form gives an exact pc for next symbol, ~~so we can use ..~~ So any kernel is usable for compression, and we can see which ones compress best. → See 99.00 ff for earlier version of kernel design.

95 We could, empirically find t kernel in following way: We simply go thru t. "coding" of t. corpus by noting a large set of bins, both sides of t. pt. in L to be predicted. Each time a char occurs, we add a "1" to all of t. bins that were t same distance from L as that / next character was. (remember we are coding in "L" space). So we end up w. the distribution in bins at distance from the no. of hits in each bin, we get t. kernel, k. | The bin allocation has to depend on N, t. no of symbols thus far.

09 We can determine empirical kernels for various texts & see if they are similar or not.

0 Also, we can measure compression of corpus A, using kernel from corpus B. This is a very peculiar measure of the distance betw. t. 2 corpi. I don't know just what width of t. kernel means;

Perhaps it would be significantly different for t. run-length encoding implied by binary codes of 2 level images of printed text. ___⌐¬___⌐¬_⌐¬___ etc.

It maybe that for most modeling of functions in Lisp-like languages that t. kernel is about t. same. Whether it would be t. same in FORTH-like langs, is unclear.

.17 AH! Since t. kernel contains all probabilistic info (assuming independence of successive choices), ~~we~~ T Kernel enables us to compute t. exact entropy, of a particular coding method.

T. Empirical kernels of 05-09 will give us t. exact compression! It may be better or worse than other PPM & PPM-like methods. Also, expect kernel to change for different types of corpi.

Lo [4.02.04] Inst Set use for finding "Phase 1": As is, t. mits I'd considered were just open loop ··· So no pgms longer than no. of symbols! The Basic pgm did have a If ( ) Then loop-structure.

OOPS can actually define functions & make recursive defns. — This inst may be too large a space so such fns.

T. types are 2 (apparently different) levels of complexity. Other forms involve t. usual: For — j = 1 to N ( ) Next & While, Until loops — The Nested If structures, etc.

Actually, Selection of INST set is part of TSQ Design.

.25 102.40 : (Mt. Carlo Lsrch): A possible way to get t. 2-D factor in speed up (as for ((Lsrch) !

As we are moving thru t. 2-3 Tree generating t. candidates, Each node corresponds to a certain (≡ suffix) prefix of the cand. — Before we go t. next Node, we inquire of t. "L" as to what t. next symbol should be: At that point is stored info as to whether all of t. pgm up to that pt. has been executed. If not, we spend our Δ of CC at that point. If finish that node & t pgm asks for another token, we get another random token & workout t. remaining cc. Then pick a new random "Cand / initialization".

0 So; T. bottom nodes ont. 23 Tree — Contain info about whether the pgms (cc) up to that pt. has been completed or not. If not completed, they give address of t. "state" of t. System at that pt., so we can continue t. work at that pt. "States" will usually be somewhat past t. pt. at which they had asked for a new token.

Mt Carlo Lsrch is essentially about t. same as ((Lsrch, but in MC, it's easier to implement t. ("computer") pickups.

In the pure PPM or PPM* "versions of proby," the pc's can be computed w. no great cc, but even then, It is probably much faster to do it Mt Carlo way.

4.01.04
4 TM

SM: On Action Algen Problem! (.11) (.29) [AAP]

0 : ON [CROSS VALIDATION]: This is one (weakest) bottle neck of t. Univ. D.f.: It is not clear how to divide corpus to employ it. — i.e. (modeling, test set) : One way: Use all possl. divisions of N into k + l : We will get (k , l) → k+l=N
a mean error measure over t. Test set for each k,l pair (assuming "Best fit for each set of models" for each k,l). So we have corresponding this mean error as a funct of k,l & we take value that converges, for l (test set size) = 0.

In many cases, it is not so easy to divide up data this way. But in some, if data is unordered, One can pick k of modeling set from t. N in $\frac{N!}{(N-k)!\,k!} = \frac{N!}{k!\,l!}$ ways. T. total no. of ways = $\sum_{k=0}^{N} \frac{N!}{k!\,(N-k)!} = 2^N$.
07 This "size" may be so large that one may want to use only part of it — using "Random Sampling". → (13)

I vaguely remember doing something like t. forgoing, but ending up w. something equivalent to using all of t. data for t. modeling set! ||
0
11 [AAP] [SN] On t. A.A. (Action Algen) problem: Perhaps Good way! Try to get t. Pd of t. A.A. over t. data. This should be directly summarizable to UDF methods. → (.29)
12
13 (.07) If random sampling is done, one might concentrate it on t. small l and. l = 1, 2, 3 etc. ... This may be a common way of using Cross Validation. (e.g. l = 1).

0:10021 More on RANDOM Lsrch: At first keep track of which Cands have been "hit" (by t. M carlo) before. — but don't yet store their states. If they are hit again within a certain time, we start storing their states. This means we don't consider Cands w. pc's below a certain level. This level will depend on how much RAM we have (assuming we don't use Disk swapping). Actually, they may be quite adequate — we may be quite happy using only 10 k most probable Cands (This sounds like a bad kind of SUMAC!).

26 As for keeping track of Cands: we already have cands in Lex order (?) — so we can have a register that tells where t. params of t. Cnds are.

28 Quick idea: That M Carlo generation of Cands could be rather cheap : but compared w. deterministic Lsrch (.34)

29 (.12) → [SM]! Say one has a strategy w. certain params Θ_0, giving yield Y_0 for t. corpus.
0
In param space, take random pts about $\vec{\Theta}_0$ & see how narrow t. $\vec{\Theta}_0$ region that gives ± Y_0 yields.

Also, given yield Y_0: How large is region in Ω of $\vec{\Theta}$ space that has yield $\geq Y_0$? what is Say it is $V(Y_0)$. Then what is $\frac{dV}{dY_0}$? — t. density around Y_0?

34 (.28) One way M t carlo would work! We have this "L" in Lex order: we start w. initial symbol, Λ — we go to that pt in L and we get a distribution of possl. following tokens, in t. "l" direction ("l" direction is "Lexical direction"). If t. corpus is of size N, then we pick a random symbol to follow, using a Kernel of width ∝ N; in t. "l" direction. for initial symbol "Λ", we can only look in t. "positive" l direction.
As we generate more symbols of t. Cand, we can look/in both more directions. Having chosen one to follow, (X say) (Λ, say) after Λ we look up ΛX in t. file L & get (probabilistically) t. next symbol, etc.
→ 103.00 but must also specify 103.25

3.31.04
4 Ray

HDD speed of Read/write

Fry's sav ~30MB/sec read or write for 7200rpm IBM 40G.

others were < 5 MB/sec. Use of caches for read & for write is crucial.

Saw some others w. ~ 40 MBy/s.. This was at least 16 MBy chunks.

WestDig Caviar WD200JB did 68 MBy/sec w. 128kb chunks (Byte or Bit?)
51 " " " " 8kb "

So assume 60 MBy/sec. 17 ns/byte. ~~...~~ = This ~ 85 clocks at 5 GHz cpu
probably. it will take > 85 clocks to do 1 token?

Feb 2004 On another web page only 10 MBy/sec write but up to 90 MBy/sec read.
(WD Raptor 740GD did 40 MBy/s in write ~ 70 MBy in read. $(\frac{1}{40}+\frac{1}{70})^{-1} \to 25$ MHz (read+write) ~ 42 ns
So, the Q is, can the disc read bytes to cpu faster than it can process them?

If, say 4 times as fast we can do swap & compute sequentially

— Otherwise, save time by doing it in ||: To some extent, this is done, if Disc has a big Cache. We have to do a read & a write to swap.

Hvr, if the instructions are at all complex, the time for read will be >> of time of insts.
In a more directly performance TM, we will be able to afford more complex Storage Systems

Another possy is to have several HDD's in || feeding the CPU: Or pick off data from several read heads in || — (More than is currently done: I saw say 16 or 32 heads in ||).

Hvr: I will probably start w. TSQT, then look into || or Random Search if it looks like the factor of 20 could be achieved or even a factor of 2 more cc!

We might arrange so that Time spent on each card would depend on the rate of which swapping occurred.

If read/write cache memys are properly used, the disc should be either reading or writing at all times. (Almost all — since read rate ≠ write rate.).
Just how the HDD system manages 4 heads read rate ≠ write rate is unclear! — i.e., it would seem that both would have to be synchronized w. the bytes going by on the disc; at a constant rate!

→ unless every other bit is written, or every nth bit is written & every l'th bit is read.

→ The No of Heads on a HDD can be very large; but normally only, say 8 (or 16?) or (32?) might be used at one time. Partly, 64 could be done w. a 64 bit CPU?
Empirically, Several recent old HDDs have 16 heads: Older ones less (15 for common!) 5, 6 ect.) Hvr 16 is the max I've seen. The 5¼" Caviars: I don't know how many heads they have.