Internal Use Only (非公開)

002

TR-IT-0096

Capturing Long Distance Dependencies from Parsed Corpora

Arian Halber

1994.12

Summary

Purpose

We are aiming at improving Speech Recognition using specific language modeling; we want to reveal Long Distance Dependencies, which are obvious to humans but completely ignored by bi- or tri- gram models.
To track *LDD* automatically, yet to keep them swift and consistent, we propose to use pre-parsed data.

Study

We propose to

- Reckon the dependencies automatically
- Use them as statistical predictors
- Evaluate their efficiency for the recognition task

The study is based on the Penn Tree Bank, a corpus of syntactally parsed data. We define two rules of *LDD*s, *Brother* and *Parent*, and extract them, along with Bigrams, from a Training set. We studied particularly the ATIS corpus, despite its small size, for its well aimed quality.
We estimate and compare Perplexities of (Bigrams+*LDD*) and (Bigrams only) models, it quantifies how much *LDD*s relieve the recognition task. We obtain roughly 8% improvement on Testing set.

Conclusion

Brothers have little influence on Perplexity; though consistent, as shown by their Weight, they are still too scarce. Parents are more common and their consistency capture Information thus improving Speech Recognition.

Acknowledgement

I‘d like to thank all the people I met in ITL department who made my stay there so enjoyable; for all the help they offered, for their ready explanations and enrapturing talks and for their attention. I didn’t know everyone well, but those I learned to know, I won’t forget.
This study would not have taken place, of course, without my supervisor, Isotani-San, along with Matsunaga-San.
Thanks also to head researcher Sagisaka-San for his precious advice, and fluent French.
And to my internship corespondent Frédéric Bimbot, for his kind interest.
Special thanks to Katoh-San who just managed to retrieve my transfered files.

Summary

Purpose

We are aiming at improving Speech Recognition using specific language modeling; we want to reveal Long Distance Dependencies, which are obvious to humans but completely ignored by bi- or tri- gram models.
To track *LDD* automatically, yet to keep them swift and consistent, we propose to use pre-parsed data.

Study

We propose to

- Reckon the dependencies automatically
- Use them as statistical predictors
- Evaluate their efficiency for the recognition task

The study is based on the Penn Tree Bank, a corpus of syntactally parsed data. We define two rules of *LDD*s, *Brother* and *Parent*, and extract them, along with Bigrams, from a Training set. We studied particularly the ATIS corpus, despite its small size, for its well aimed quality.
We estimate and compare Perplexities of (Bigrams+*LDD*) and (Bigrams only) models, it quantifies how much *LDD*s relieve the recognition task. We obtain roughly 8% improvement on Testing set.

Conclusion

Brothers have little influence on Perplexity; though consistent, as shown by their Weight, they are still too scarce. Parents are more common and their consistency capture Information thus improving Speech Recognition.

Résumé

Objectif

Nous cherchons à améliorer la reconnaissance de la parole par une modélisation spécifique du langage. Nous désirons mettre en évidence des Dépendances Longue Distance (LDD), dépendances aisément identifiées par l'utilisateur humain, mais complètement ignorées par les modèles de bi- et tri- grams.
Afin de détecter les LDD automatiquement, tout en garantissant leur souplesse et leur cohérence, nous préconisons d'utiliser des données pré-étiquetées.

Étude

Nous nous proposons :

- de comptabiliser les dépendances automatiquement
- de les utiliser en tant que prédicteur stochastiques
- d'évaluer leur performance aux vues de la reconnaissance.

L'étude se fonde sur le *Penn Tree Bank*, un corpus de textes organisés en syntagmes. Nous définissons deux règles de LDD, Frères et Parents, que nous extrayons, ainsi que les Bigrams, à partir d'un corps d'apprentissage. Nous avons particulièrement étudié le corpus *ATIS*, en dépit de sa taille restreinte, pour son contenu très bien ciblé.
Nous estimons puis comparons les Perplexités des modèles (Bigrams + LDD) et (Bigrams seuls), afin de quantifier l'allégement de la reconnaissance du aux LDD. Nous obtenons environ 8% d'amélioration.

Conclusion

Les Frères influencent peu la Perplexité; quoique cohérents, comme l'indique leur Poids, ils souffrent trop de rareté. Les Parents eux, sont plus répandus et leur cohérence capture de l'Information, améliorant en conséquence la reconnaissance de la parole.

Les mots sont si vivants, j'ai l'impression parfois qu'il ne leur manque que la parole.

(bavardage)*

---

* words are so lively, I feel sometimes they only lack speech. (chatting)

CapturingLong Distance Dependencies from parsed corpora

TABLE of CONTENTS

1. Introduction

1.1 Project Context

1.1.a) ITL Project

Interpreting Telecommunication chain

The ITL project consists in carrying out a multi modal multi lingual system, oriented toward meeting and conference planning. The project, conducted in cooperation in several laboratories across the world, involves the general chain:

speech in foreign tongue -> recognition -> translation -> synthesis -> speech in Japanese

1.1.b) Study

prospective speech recognition

Our aim is to improve the speech recognition stage. The study is conducted on an English data base its application in ITL project would be on Japanese. Mainly it's a prospection trial and evaluation of new means in speech recognition.

1.2 What is speech recognition?

We'll understand it as the *estimation of a word string given an incomplete knowledge on this string*. for example given the spectral wave forms of an acoustic input, or given a string of phonemes or given neighbor strings etc.

Incompleteness of knowledge, which fosters *uncertainty* [2], may come from different factors:

- channel noise
- pronunciation
- lexical choice
- syntax
- semantic

Guessing a string while taking in account each of those directions, is the key in solving the different uncertainties. We will perform our guess by assessing the strings' probabilities.

1.3 Linguistic approach

We'll deal with the last 3 factors previously enumerated, i.e. the linguistic freedom, or we may say *information*[1], of the speech vector.

The field of these factors is quite close from the aim of recognition, i.e. the words, and if uncovered, has much *prediction power* : Listening to a discourse, we have all the more facility making out the words as we have a clearer knowledge of

- the tongue
- the subject
- the context;

Of course, we need to know the vocabulary or just can't "recognize" it, but in a dynamic way, we're using all the previous words in the string to deduce the next one. If humans perform this task very naturally, the recognizer has to learn the ways words work together and "call each other" in the sentences. We'll see these intuitive aspect in more detail as we expose the basics of speech recognition using language models, in part 2.

Now, prediction deduced from previous linguistic information is necessarily bounded; however well we may understand and use words' interactions, we can't guess everything that will be said, that would negate the fact that some information is brought up, by speech. Ideally we want to recognize only this non redundant information. Thus, our effort is to come closer from that bound, so that final recognition is alleviated as much as possible. We conduct our study according to that view which belongs to Information Theory.

1.4 Toward a general model

Inspired by human performances, and to reach a most general yet handy modeling, We'll propose a new language model, *long distance dependencies*. We'll expose its definition in part 3, then describe in part 4 an implementation of the model in a recognition task, we'll expose its results and evaluation in part 5. We'll draw conclusions and propose further developments in part 6.

2. Language Modeling for Speech Recognition

2.1 Bayes Law

2.1.a) Speech recognition process

We can modelize the speech process from utterance to recognition as follows;

- A words string is uttered W0(w1, w2,..,wn) from a finite vocabulary.

First uncertainty; we don't know what string is uttered, this is modelized by P(W0)

- It gives rise to an acoustic wave form A

Second uncertainty; we don't know what wave form is created, given the word string, this is modelized by P(A/W0)

- Using pattern recognition, with A as input information, we chose a candidate W'(w'1,..)

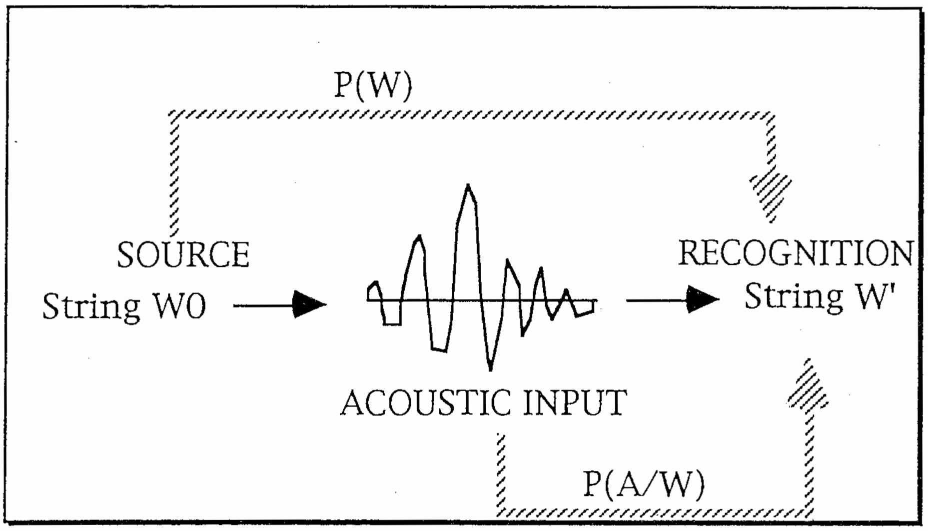


figure 2.1.1

The choice of W' is uncertain because of the two previous freedom. Let's ponder more closely on that choice.

2.1.b) Decision criterion & Bayes' Law

We're using the most likely candidate, Maximum Likelihood estimate is efficient, consistent, unbiased and simple. Most likely candidate W'= most likely string to have given rise to A

$$P(W'|A) = \max_{W} P(W|A) \tag{2.1.1}$$

We introduce Bayes law:

$$P(W|A) = \frac{P(W)P(W|A)}{P(A)} \quad (2.1.2)$$

Applied on (2.1.3) it entails

$$\Rightarrow P(W'|A) = \max_{W} \frac{P(W)P(W|A)}{P(A)} \quad (2.1.3)$$

A is fixed, so that it comes down to:

$$W' = \arg\max_{W} \; P(W)P(A|W) \quad (2.1.4)$$

We can distinguish two terms of quite different essence:

- **acoustic matcher** [2][7] term $P(A|W)$; this is due to the variability of wave form according to the string that gives rise to it. This term can be estimated using an acoustic matcher on a training set, this won't be the object of our study.
- **language model** term $P(W)$; This is the distribution on the source - RE: Information theory - in most cases we can't get directly to that value for the given language, so that we'll have to use an estimate instead, which comes down to substituting the language by a more or less explicited model.

2.1.c) Using language model: history and classes

What do we mean by language model?

c.1) Estimating language distribution

Our aim is to estimate the "language model" term that is to say;

$$P(W) = \prod_{i=1}^{n} P\left(w_i \middle| \underbrace{w_{i-1}, w_{i-2}, .., w_1}_{history}\right) = \prod_{i=1}^{n} P(w_i|h) \quad (2.1.5)$$

But the event space of histories and words *(h,w)* is too large, and no reasonable amount of data would be sufficient to span it. If we take in account history in its totality, the increase of parameters is exponential, for evident scale problems it is impracticable, even dealing with very limited vocabulary and string lengths That's why we need a model, i.e. simplifying assumptions.

c.2) Clustering

We have recourse to equivalent classes. A mapping S of the event space h is defined: histories that fall into the same equivalence class are supposed to a same effect on the probability distribution of the next word w,

$$P(W) = \prod_{i=1}^{n} P(w_i|S[w_1, \ldots, w_{i-1}]) = \prod_{i=1}^{n} P(w_i|\bar{h}) \quad (2.1.6)$$

The idea is to select relevant data from history. Namely, words that have an expected influence on the next occurrence, are kept, and all other words in the history are discarded. There are different families of clustering, it may be,

- knowledge based, statistic[2]
- supervised, unsupervised[1]
- defined, iterative[11][5]
- general, adaptative[5]

Ideally the model should be a mixture of all methods.

c.3) Bigram example

The partition of histories is based on the last word of the history, the underlying assumption is Markovian;

$$P\left(w_i \middle| w_{i-1},\ldots,w_{i-n}\right) \cong P\left(w_i \middle| w_{i-1}\right) \qquad (2.1.7)$$

The model makes profit with such dependencies as

next time
'd like
loathed enemy

Statistical implementation of Bigrams involves a physical clustering of the training text; A two-words wide window is put, and then slided, on the text, so as to reckon the concerned two-words sequence. We collect thus information on the sequence's probability in order to build the following model:

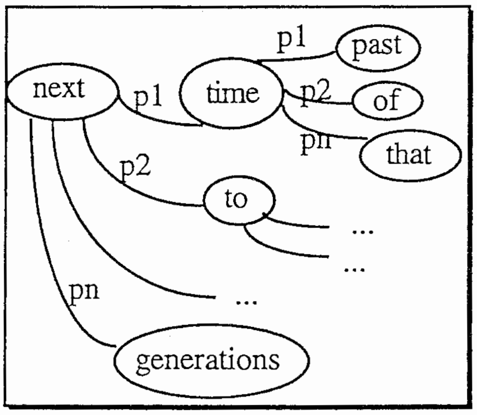


figure 2.1.2

where **pi of different levels are independent.**

For example, we estimate in ATIS corpus that,

$$P\left(w_i = like \middle| w_{i-1} = 'd\right) = 0.62$$

To use this conditional probability we assume that whatever the words preceding "*like*"

...if possible I'd like...
...that is why I'd like...

its probability to occur next is not affected.
It seems reasonable enough. Yet the first example may appear more natural because of the preceding *'if'*, we sense this relation, but Bigram is blind to it, as would be any N-gram model, at best sensitive to *'if possible I'd'* if a 4-gram, but sparseness limits drastically high level N-grams (cf. 2.3.c).

Then, to what extent is the Bigram assumption correct? Actually, what we're interested in is its efficiency in the recognition task's respect. Information Theory can give us a clue.

2.2 Information theory

2.2.a) Language as a stochastic source

a.1) what is language?

Our notion of "language" is not reduced here to a tongue or a vocabulary notion, it includes the frame and use of the speech; '*an English colleague planning a rendez-vous on the phone*' or '*travelers asking for flight schedules in US*' or '*dada Poetry* ' .
So we define language as the concordance of

- a finite vocabulary Set V
- a context of speech

Given that, a language defines a probability distribution on word sequences.

a.2) Information Source

We can thus consider language as a source of Information and words as its outputs ; words are put out according to the aforementioned probability distribution. Concept of Information Source is intuitive enough, it might be added that *observing an information source* is the exact equivalent to *running a random experiment.*, so that, from a probabilistic point of view, information source can be seen as a generator of random experiments.

Owing to probabilities, the user has uncertainty about the identity of the coming word. This *uncertainty* is related to the novelty, the *information*, conveyed by this word; uncovering a word is all the more difficult, i.e. uncertainty deeper, as its information content is high. Actually, *uncertainty* becomes *information* as soon as the output is discovered by the user, they are the two faces of the same coin. Let's define further this *uncertainty / information* concept.

2.2.b) A definition of information: *entropy*

How to quantify Uncertainty / Information? We allege it is function of the outputs' probability distribution, but what function is appropriate?

b.1)intuitive approach

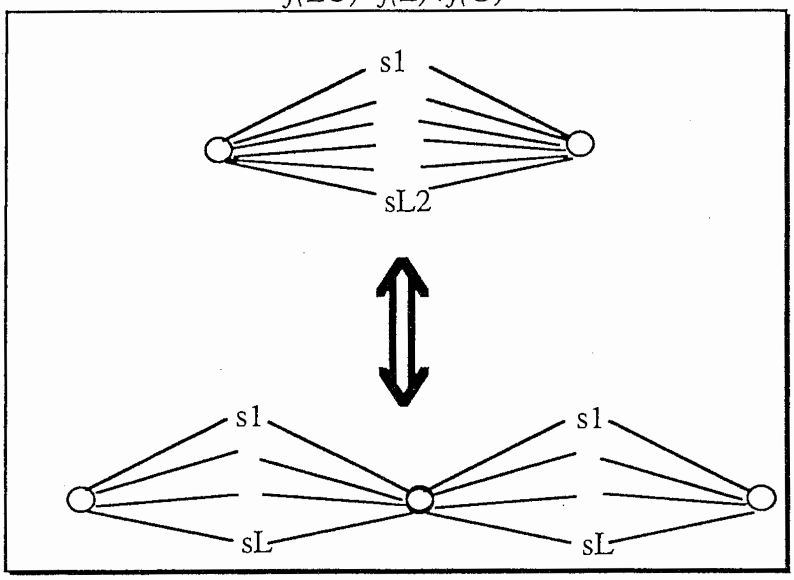
Given a source S with a set V of L symbols, uncertainty about the next output, therefore information, is maximal if each of the possible symbols are chosen with equal probability 1/L and independently of previously chosen symbols.
The information content / amount of uncertainty of such a source is:

$$H(L) = \ln(L) \qquad (2.2.1)$$

It is the only form of function to abide by the following four natural proprieties;

i) A measurement of the amount of uncertainty involved in S should be a function $f(L)$ of L.
ii) Since there is no uncertainty when S has one possible outcome, one should have $f(1)=0$.
ii) In addition, the larger L, the larger the uncertainty involved in S, so that $f(L)$ should be an increasing function of L.
iv) Let T be another uniform independent information source, and consider the new source ST putting out joint observations $(w_S w_T)$, with $L*G$ outcomes occurring with the same probability $1/L*G$. Assume that S and T are independent. Given those conditions one may expect that information involved in S and in T add, when measuring information of ST, or equivalently that uncertainties are summed up;

$$f(LG)=f(L)+f(G)$$



2 times L or L^2 : two equally difficult tasks

figure 2.2.1

In particular, as illustrated in *figure 2.2.1.*, enlarging the vocabulary L up to L^2 is equivalent to expecting pairs of words instead of singles. Intuition tells uncertainty is doubled and not squared, this aspect is called the branching factor[3].

The only function which satisfies the assumptions (i-iv) above is

$$f(L) = k\ln(L).$$

for k=1, it gives back (2.2.1) .

b.2) general case

In general cases, probability is not uniform. Let w denote a symbol put out by the source with probability $P(w)$, then it can be shown that the proper measure of information / uncertainty is entropy H :

$$H = -\left[\sum_{w=1}^{L} P(w)\log P(w)\right] \tag{2.2.2}$$

The uniform independent case exposed in b.1) appears as a particular case of this formula. Moreover, the general source has then as much information content as a uniform binary source of size

$$L' = 2^H \tag{2.2.3}$$

We will discuss in coming *b.4)* and *c)* interpretations and consequences of this equivalence.

b.3) calculating Entropy

How can one calculate entropy of a given source?

a way to reach (2.2.2), on a sequence of outputs w_i , is,

$$H = -\lim_{n\to\infty}(1/n)\left[\sum P(w_1,\ldots,w_n)\log P(w_1,\ldots,w_n)\right] \tag{2.2.4}$$

Then, assuming ergodicity,

$$H = -\lim_{n\to\infty}(1/n)\left[\log P(w_1,\ldots,w_n)\right] \tag{2.2.5}$$

For a very large corpus of speech or text, we consider the sequence's length as infinite, so that

$$\boxed{H = -(1/n)\left[\log P(w_1,\ldots,w_n)\right]} \tag{2.2.6}$$

Thus Entropy can be estimated from a long sequence of symbol.

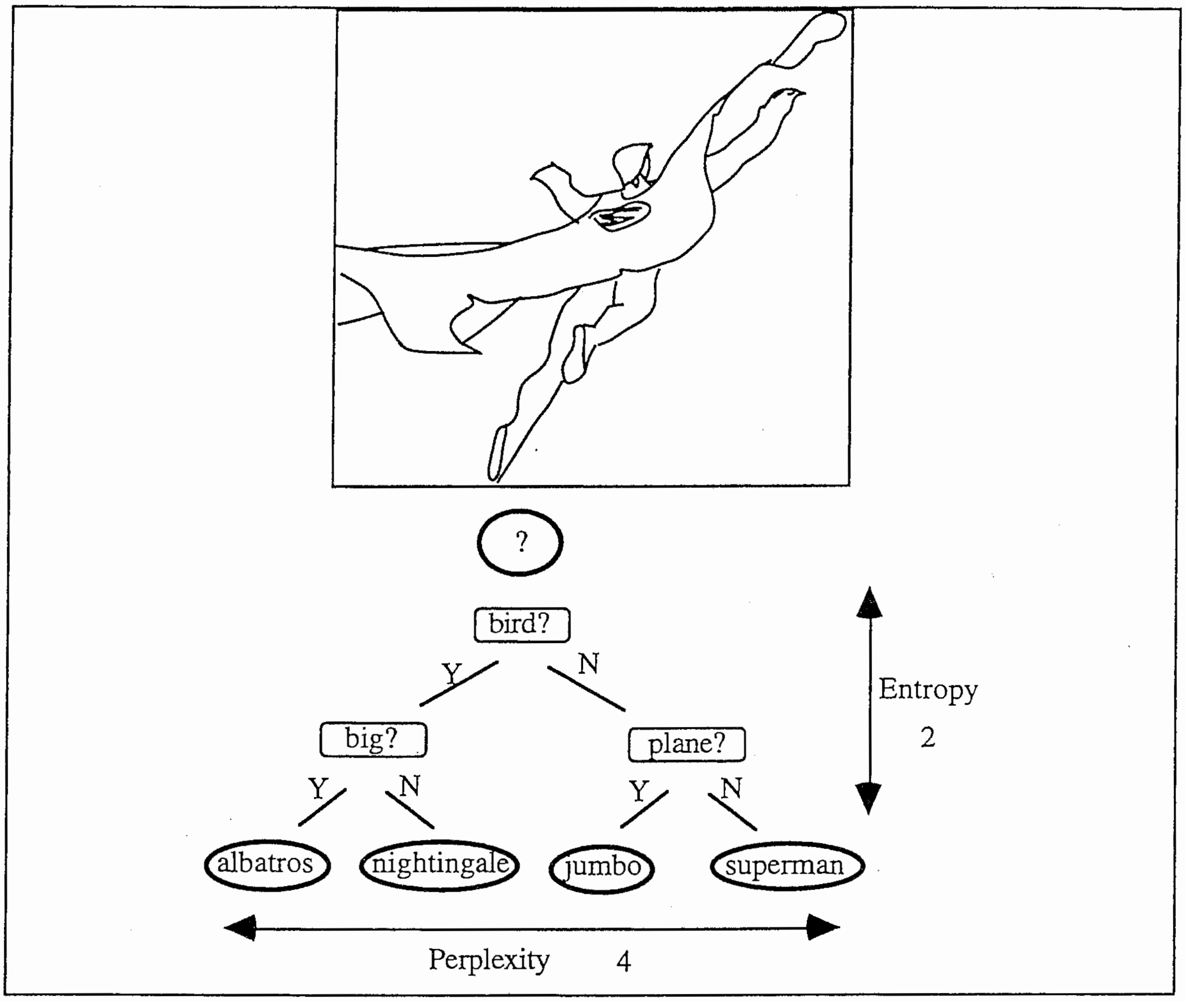
b.4) interpreting Entropy

- *as an information content ;*

According to Shannon's theorem, any encoding of the source must use at least H bits per word, on average; this theorem is in accordance with (2.2.3), as the uniform binary source with the same information content as S requires H bits, and this equivalent source maximizes information due to its uniform distribution.

Said another way, an output will reckon H bits of innovation; those bits can't be deduced from the former outputs, whatever the coding the behavior attached to S.. They figure the non-redundant information of the output.

• *as an amount of uncertainty ;*



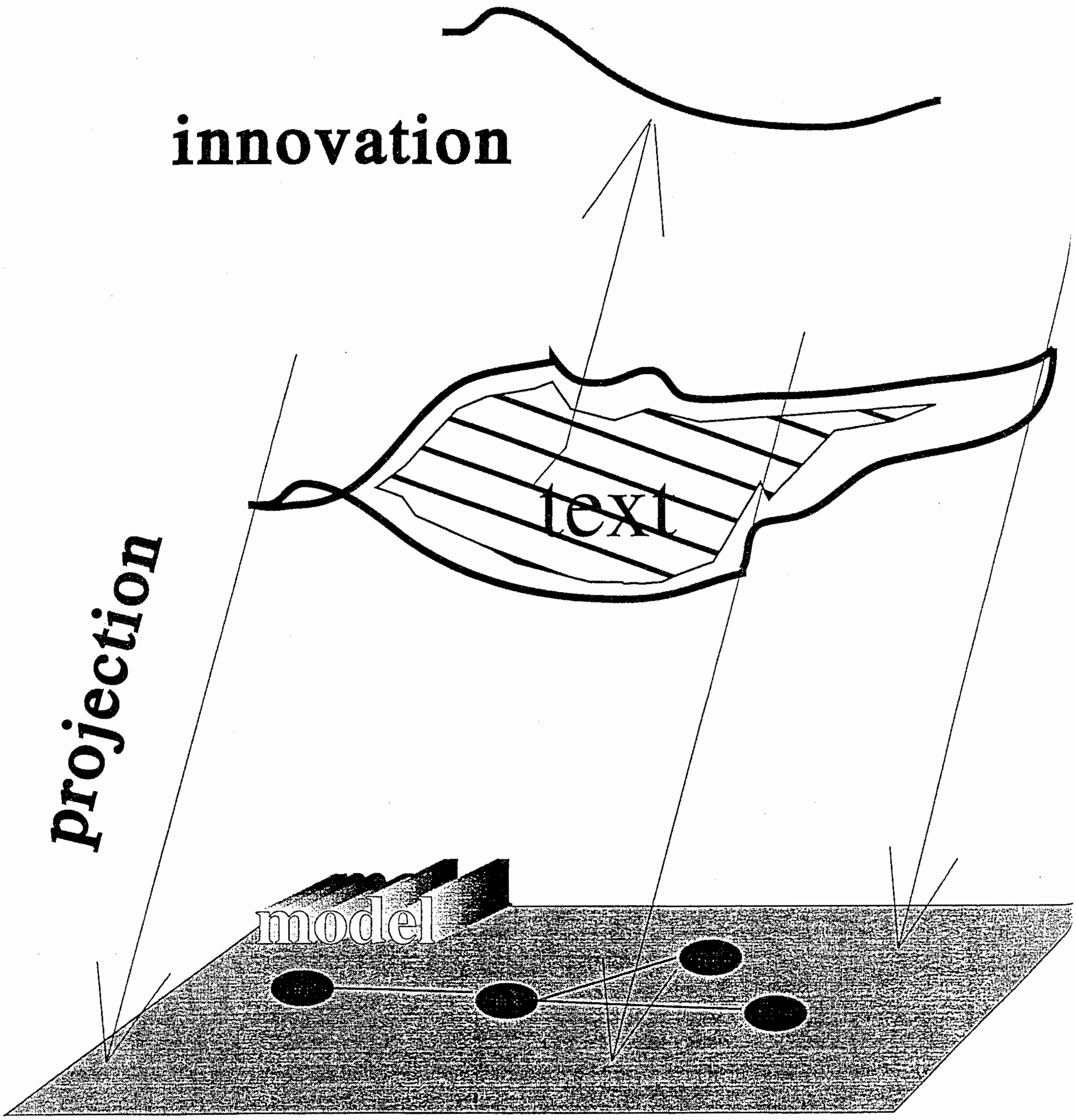
amount of uncertainty:

'is it a bird? is it a plane? No it's Superman'

figure 2.2.2

H is an estimate of the recognition difficulty of speech generated by the same source. The irreducible H bits stand for the unavoiding H "yes/no" questions to get to the identity of the output, this is illustrated in *figure 2.2.2* This branching of questions lead to $L'=2^H$ word candidates, whom the Source puts out uniformly, so that the user, how smart he may be at guessing the source's behavior, has no choice but to discover the chosen candidate among the list once put out. The average number of remaining candidates is the source's intrinsic *Perplexity.* ,

$$\boxed{PP_{intrisic} = 2^H}. \qquad (2.2.7)$$



speech recognition

language model : **projection**

remaining entropy : **innovation**

2.2.c) Perplexity, an evaluation

c.1) Perplexity of a model; projective view

In real case, expression (2.2.6) is not known exactly because the probability of a word sequence is not. Hence the approximation by a language model, as already exposed in part *2.1.c)*. What is calculated then is no longer the Entropy but a projection $\hat{H}$ expressed as the *Logprob*,

$$\hat{H} = LP = -(1/n)\left[\log \hat{P}(w_1,\ldots,w_n)\right] \qquad (2.2.6)'$$

Why do we call it projection? From the standpoint of original definition (2.2.2), Logprob stems from a*cross Entropy,*

$$\boxed{\hat{H} = -\left[\sum_{w=1}^{L} P(w)\log \hat{P}(w)\right]} \qquad (2.2.2)'$$

following the same scheme as in *b)*, *Perplexity of the model* is defined,

$$\boxed{PP_{model} = 2^{\hat{H}}} \qquad (2.2.7)'$$

Perplexity is the source's Perplexity viewed from the language model. This is illustrated in *figure 2.2.4*. It can be demonstrated easily[3] that

$$H < \hat{H} \qquad (2.2.8)$$

Therefore, following (2.2.7), 2^H is an inferior bound of any model's Perplexity

c.2) Perplexity, difficulty of the recognition task

Following *b.4)* , Perplexity of a model expresses the average number of unsortable equiprobable candidates, after the model was applied on the source. Thus it evaluates the difficulty of the recognition task. If we adopt that view, Perplexity becomes an evaluation of the model's efficiency regarding recognition; how well does it capture the source's behavior - i.e. regularities - in order to alleviate final recognition ?

To sum up:

The user assumes a certain behavior for the source; a model. Given those assumptions and given the previous outputs, he tries to guess the next output. If the source is not determinist he can't guess accurately, to complete his guess he still need *some* information on the output. This amount of information is *entropy* $\hat{H}$. It is supposed to solve his *perplexity* concerning the output, i.e. his remaining hesitation between PP equally possible outcomes. This requires $\hat{H} = \log_2 PP$.

- The smartest user on earth has an average perplexity per output,

$$PP_{intrisic} = 2^{H}$$

where
$$H = -\left[\sum_{w=1}^{L} P(w)\log P(w)\right]$$

- The user simplex has an average perplexity per output,

$$PP_{model} = 2^{\hat{H}}$$

where
$$\hat{H} = -\left[\sum_{w=1}^{L} P(w)\log \hat{P}(w)\right]$$

A model is all the better than it lessens Perplexity

c.3) application ; the Shannon game

C.E. Shannon invented that game in order to estimate entropy of English. It consists in measuring perplexity of a human confronted with a text to discover. He tries to guess a letter and is told about the correctness of his guess. when he has found the right answer, he passes to the next letter. As in *figure 2.2.2* the average number of guess he makes to uncover a letter equals the entropy per letter of the text, seen from his model. His model is implicitly the way he conducts his guesses. The experience of Shannon in 1951 resulted in an entropy of 1 per letter, which means 1 bit is sufficient to code a letter

2.2.d) Joint sources: mutual information

d.1) definition; a loss of entropy

It is very useful to have a measure of information provided by outputs symbols x of a source S about output symbols y of a related source S'. Typically, y is some extracted component of S ; whose knowledge is used by the language model, for example, $y(i)=x(i-1)$, output preceding x..

According to the information / uncertainty duality exposed in *b)*., mutual information is a loss of uncertainty, i.e. a loss of Entropy. Applying (2.2.2),

$$\begin{aligned} I(X|y) &= H(X) - H(X|y) \\ &= -\sum_{x=1}^{L} P(x)\log P(x) + \sum_{x=1}^{L} P(x|y)\log P(x|y) \qquad (2.2.8) \\ &= \sum_{x=1}^{L} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \end{aligned}$$

Globally, the loss of Entropy achieved on the source S by the knowledge of S' *is*

$$I(X;Y) = H(X) - H(X|Y) \qquad (2.2.9)$$

It can be thus calculated,

$$I(X;Y) = \sum_{y=1}^{L'}\sum_{x=1}^{L} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \qquad (2.2.10)$$

- Particularly, contribution of a joint occurrence (x,y) can be expressed as,

$$P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \qquad (2.2.11)$$

it can be negative, hence counterproductive, if,

$$P(x,y) < P(x)P(y)$$

that is to say, if (x,y) occurrence is an accident. This remark simply means that observing accidental (x,y) is misleading and hampers deduction of X from Y. The more regular the occurrences, the richer the mutual information. In an extreme case, if the two sources have similar behavior, the Mutual Information is sufficient to uncover S; it is then equal to Entropy.

- It is symmetric in X and Y as suggested by the denomination "mutual"; The amount of information provided by S' on S is the same as provided by S on S', it reflects the way S and S' are correlated.

d.2) application; predictive power

The user assumes some information on the source as known ; for example some preceding outputs. The model defines which information are considered and how they are used to guess the next output. Therefore, it is quite useful to assess the potential prediction of different information sources, as bigrams (the latest output), trigrams (the latest two outputs), or the latest but one output , or the latest function word etc.

Mutual Information provides an atomic view of this potentiality, through expressions like (2.2.11);

$$I(x=\mathrm{I};y=\mathrm{would}) = P(x=\mathrm{I},y=\mathrm{would})\log\frac{P(x=\mathrm{I},y=\mathrm{would})}{P(x=\mathrm{I})P(y=\mathrm{would})} \qquad (2.2.12)$$

If S' is supposed to generate the latest word, then (2.2.12) may equal a high value (note that if S' is supposed to generate the two but latest word, (2.2.12) may very well be negative).

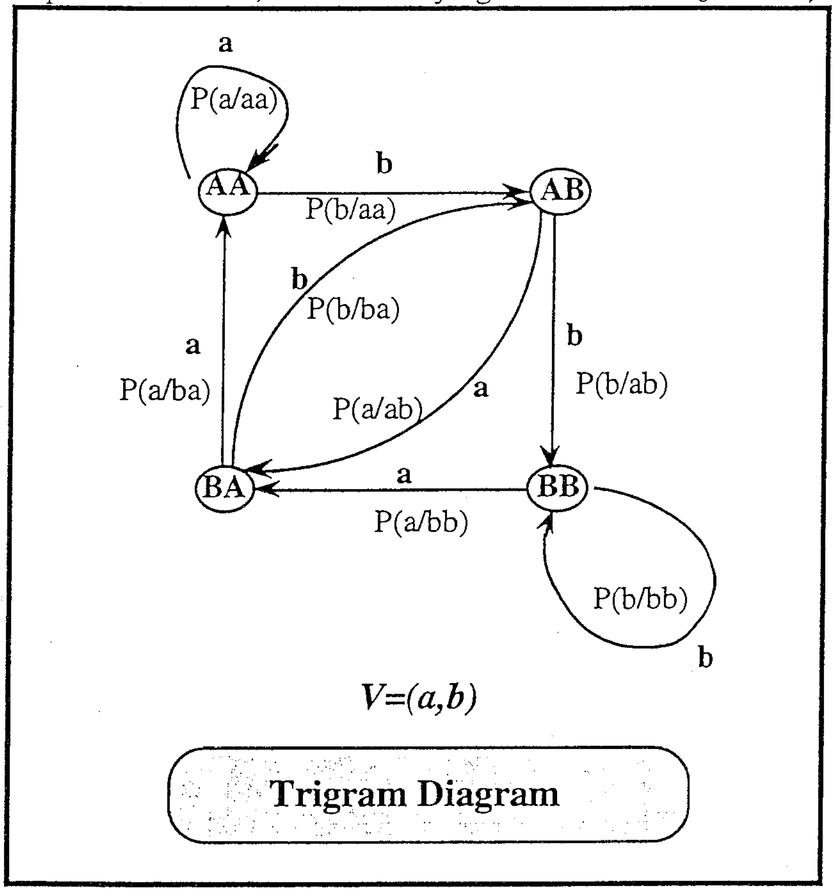
Mutual Information stands as a first approach of the modeling power.

Summing up global mutual information on the vocabulary, actually estimations on long strings of words, yields a possible lessening of entropy. Yet the lessening is widely overestimated because is doesn't take in account cross correlations between words, just adding blindly.

2.2.e) N-gram example

e.1) language as a Markov chain

The Source is supposed to generate a chain of state, whose transitions depends from the departing state only. A transition puts out a symbol. Thus the Source's behavior can be reduced to a state's behavior. Assume that state S_i is the collection $(w_{i-1},..,w_{i-N})$ it expresses that the next output depends from the preceding N outputs. This is the equivalent of the Markovian assumption referred to, about classifying and clusters in § *2.1.c.3)* .



trigrams: four states of a binary source

figure 2.2.3

e.2) mutual information of neighbors

To assess the validity of the model, we estimate such expression as (2.2.12) for bigrams, or replacing y by $y_1..y_{N-1}$. Of course we have only estimates of such expression, as well as global Mutual Information. estimates calculated on a training set. Doing so, we estimate the amount of information involved by neighborhood occurrences of words.

e.3) remaining entropy

The "neighbors' information" applied on a text, achieves a projection of the text on the model structure - here the Markov chain. The projection leaves a remaining component; consisting in what can't fit in the model. It is the innovation of speech -or text- regarding the model, i.e. entropy viewed by the model, as illustrated in *figure 2.2.4*

As repeated much now, this entropy can't be curbed under the intrinsic entropy H which is unknown . Yet there is strong assumption that models currently used fall far off from the ideal model. An hint is given by the Shannon game, exposed in *c.3)*, and which consists in comparing human unconscious model and trigram models or as such.

An assessing of this "remaining component" is given by entropy, or by perplexity; the difficulty of a recognition based on the N-gram model.

Now, there are two limitations to this information;

- the intrinsic innovation of the text.
- the information included in further remote outputs.

2.3 Automatic modeling

Now we discussed the need and use and assessing of language models let's expose briefly ways to spawn such a model.

2.3.a) Using expert knowledge

The model is asserted by linguist expert, for example defining syntactic or semantic rules.

a.1) good points:

- Straight forward; the model is given, no iterative computing is needed
- Global; features retained by linguists are generally invariant for a given language so that it fits lots of context
- Gain of time and parameters; it is not so parametrical dependent as automatic or unsupervised methods.

a.2) drawbacks:

- Only discreet criteria; the knowledge is generally a yes/no answer or a ranking, not a distribution.
- Insufficient expert knowledge; experts' model are still too raw or to general to be efficient directly on a recognition task
- Variability of language in time and space; expert knowledge is bound to change, according to time periods, and most evidently according to tongues.

a.3) Example ; Tagging

A common expert knowledge used in speech recognition is the tagging i.e. a classifying of words or word groups under a label. the Label may be syntactic, or a grammar tag, or more generally a Part of Speech; noun, verb...

2.3.b) Using statistics

b.1) why statistics ?

Using statistics in Speech recognition is very natural, we could them emerging because of uncertainties in the speech recognition chain in § *2.1*.

They allow a continuous recognition, yielding probabilities instead of discreet choice. Moreover they can be used on data for training hence the better adequacy and robustness of the recognition.

b.2) context of statistic modeling ; training and testing sets

Parameters are extracted from a statistic analysis of a corpus. The parameters characterize a model. In some case the model is assumed and parameters just fit in, in other cases, the model along with the statistic analysis, yet it as some original form.

The training data has to be large enough

- to achieve the ergodic assumption, and if not really, at least the following conditions,
- to be a truthful image of the model
- to collect maximum cases and occurrences

The model is conditioned to work on data that have the same behavior as the training data ; The stochastic source we want to guess must be the same as the one who gave rise to the training data.

To evaluate the model, we run it on a testing set which abides by the preceding constraint.

Now, a model is never built only from expert knowledge, statistics are used to complete and adapt the linguistic assumption. Nor is it only made from statistical data, or what parameters are we looking for? A merging of the two approach is needed. .

2.3.c) N-gram example

Once the Markovian assumption is adopted, the parameters of the model are deduced statistically from a training corpus.

c.1) number of parameters ; first limitations

Given a vocabulary of L, let us figure out what scale reach an N-gram modeling. For example et L=100 words; this is quite a restricted vocabulary, so we're in a very optimistic hypothesis.

- there are 100^2 =10 000 possible bigrams. Even assuming lots of them don't occur;
  - the number of resulting parameters is still around 10 000

- a very large training corpus is required to meet at least once the occurring bigrams

- there are 100^3 =1,000 000 possible trigrams, the preceding scale problem is much worsened, mainly, no amount of data can collect all existent trigrams, not even a reasonable rate of the existent trigrams. This problem is called sparseness, it is due to the basic inadequation of a parametric model ; our statistic estimation, to a non parametric phenomenon ; speech generation.
- there are 100^4 =100,000 000 possible 4-grams, at this stage the sparseness is so bad we can't reach reliable estimates of 4-grams probabilities. Moreover, even if we had this knowledge, the number of parameters to handle would be too much of a burden. As a result performance of 4-grams can be only slightly superior to trigram-grams, and at a very high cost.

At this rate we don't need going on with 5-grams or higher level-grams. Sparseness and handling conditions doom any N-gram over bi- or tri-grams.

c.2) structural bound ; second limitation

• *segmental problem*

As described in *2.1.c.3)*, N-grams are collected by sliding an N-wide window on the text (or speech transcription) , this method segments artificially the data and raises problems of non contiguity, even on a local scale;
for example the sequences

booking return ticket

and *booking ticket*

have nothing to do with one another according to the N-gram window.
Some refinement can be proposed as the search of a head word[1][2], but it

- is not very robust - notion of head word is not so regular - yet there are both unsupervised and expert method to track them.

- involves a heavy additional processing, but after all that is any statistical approach's lot)

- entails a rewriting of the probability distribution and an armful of new parameters, this aspect can be rather positive, thinking of it as a new, more adequate language model.

So, it may be worth it, but note that it's already a modification of N-gram model to go beyond their particular limitations.
Now, the sequential feature of N-grams fundamentally ignores non-contiguous information, as symmetries, repetitions and alike.

• *narrow scope*

A characteristic of N-grams, actually the essence of the Markovian assumption that gives rise to them, is the oblivion of any event except for a very short term memory ; it was just pointed out in *c.1)* that N doesn't excess 4.

Even short length syntactic or semantic regularities are necessarily ignored by the model, let alone farther away information ; obviously lexical information contained in the past is incorporated only weakly in the model.

The question is now how can we go beyond the N-grams limitations we just exposed ? Improved Markovian modeling is developed through the non-linear Hidden Markov Models, which figure an alternative way to clustering and "explicit" modeling as exposed in *2.1* . Still assuming the source as a Markov process, states are considered now as outputs, so that an extra-layer is laid on the model. Needless to say the model includes a lot of parameters -because of this extra-layer which is positively "hidden"- whose estimation requires some expensive computation and algorithm.

In the field of linear language modeling, we can however imagine more general models that would take in account the lacking aspects of N-gram.

3. An Other History

3.1 Long Distance Dependencies

3.1.a) Intuition of LDD

a.1) toward an adequate and general model

Recalling formula (2.1.6) in we're looking for a adequate classifying on histories.

• *adequate?*

As close as possible to the ultimate model which yields only intrinsic perplexity when performing recognition by prediction. A more general modeling taking in account more regularities of the source should achieve a lessening of perplexity.

• *general?*

A model able to capture information *both local and global* [4] .
An efficient model to capture local constraints exists ; the bi- (or tri-) gram model.
Let us find now a model able to capture long distance constraints, ignored by bigrams.

a.2) learning from humans : Shannon type Game

The original Shannon game was exposed in *2.2.c.3)*, variations on that game were imagined, and experimented at IBM, for assessing models potentials. This time the user is given an additional source of information while guessing the text, for example bigrams, or trigrams. The comparison with the unassisted recognition provides a lower bound of the source's information. Actually, as human tends to be smarter than known modeling, it's rather an upper bound of the models performances with that source. It was observed during those experiment some clue about human's outsmarting models; mainly, the human user shows a better understanding and using of global context, lexical, syntactic, semantic. So it seems, what models need is a better approach to far relations between words. Hence our research on **long distance dependencies.**

3.1.b) Alternatives

b.1) existing models

• *fixed distance*

Obviously , fixing the distance a priori is inadequate (R. Rosenfeld[2]'s experiment on hysteresis bigrams is eloquent). Now some learning may produce some "characteristic distance" attached to a word, but it reduces the dependence to a lone distant gram.

• *triggers*

As exposed in his thesis by R. Rosenfeld[2]; this time, not characteristic distance, but characteristic words are extracted, based on an Entropy minimization. But it involves a large amount of data and computations to be relevant.

• *Multigrams*

As exposed by Frederic Bimbot [5], it consists in adapting N-gram's degree to the context, it achieves a significant fall down of parameters and allows some distant information to be taken in account. Yet it still involves contiguity.

• *function & content words*

As exposed by Ryosuke Isotani[4]; since words can be classified in Japanese into those two classes (function words often act like post-positions), the model takes in account the last function and last content words, this allows to capture separately semantic and syntactic dependencies, yet the scope is still limited - and the model doesn't transpose easily to another tongue.

b.2) charges

• *distance free*

To remain as general as possible and to be able to adapt to the context.

• *stochastic*

For a continuous recognition, and to follow the idea of adaptivity.

• *yet computationally light*

Because time is a key in speech recognition, all the more true when coupled with telecommunication.

A way to respect those constraints is to introduce an exterior knowledge into statistic analysis, to guide the analysis in a straight forward manner.

3.1.c) Choice of linguistics

c.1) a knowledge

We can use classifying, ordering and interpreting borrowed to semantic, syntax, grammars etc. This knowledge is supposed to contend information -if the expert is not totally wrong. It is thus a potential lessening of the text Entropy.

c.2) a dimension

Linguistic knowledge can be viewed as a additional axis in the text representation, whereas raw text is a linear sequence of strings. Of course, the "raw text" is not a monodimensionnal space, since it is embedded in information and correlations, but that sort of dimensions are implicit and revealed only by repeated experiments and statistics.

Linguistic axis is explicit and easy to interpret. It guides us for searching relations between words independently from the distance. It is all the more obvious than the linguistic knowledge wields classes and hierarchy in words sequences.

c.3) a consistency?

We said linguistic knowledge is easily interpreted, if our search is based on that knowledge we suspect interpretation could follow somewhat naturally, as if inherited.

Now "easily interpreted" doesn't mean "consistent". Only experiment can find consistency. But interpretation might be a good *a priori* approach of consistency.

3.2 Definition : parsed corpus & rules

3.2.a) Parsed corpus : three types of information

```
( (S    (NP   *)
        (VP   Show
              (NP   me)
              (NP   (NP   all
                          the nonstop flights)
                    (PP   (PP   from
                                (NP Dallas))
                          (PP   to
                                (NP Denver)))
                    (ADJP early
                          (PP   in
                                (NP    the morning)))))
  .)
)
```

a sentence from the parsed corpus - ATIS file

figure (3.1.1)

The parsed corpus we're working with is extracted from the Penn Tree Bank, a data base originated in a Pennsylvania university project. This corpus is interesting for his tree structure quality. It is an explicit illustration of the above mentioned *dimension* point of linguistic knowledge.

Let's explore the components of this structure.

a.1) tags

ADJP, PP, NP, VP...

(Adjective Phrase, Prepositional Phrase, Nominal Phrase, Verbal Phrase...)

There are 14 different tags, including a "unknown category" tag.

They are derived from the "Part Of Speech" labels. The latter are a mapping of words on the grammatical field - hence a potential projection of entropy. Now, the tags are not usual POS but a limited set; and they are no longer attached to a word but to a *phrase*.. A phrase is a syntactic whole ; a branch of the syntactic tree as developed in *a.3)* .

a.2) words entities

• *general aspect ; instanciates*

They are the basic information, the parsing is done out of the words data. Once done, words appear as instanciates of the syntactic skeleton. It can be observed then they appear at the ends of syntactic derivations.

• *particularities ; null elements and word clusters*

Some null elements were added in the parsing to keep the phrase's structure as general as possible. For example, '*' means stands for the understood subject of infinitive or imperative. How shall we treat the null elements? As we're interested in the skeleton of the sentence, we will keep those null elements, though artificial, in the model.

Some word clusters remain, e.g. '*return ticket*' or '*one-way flight*', due to the removing of some phrasal nodes, found both problematic and inconsistent for the tree parsing.
How shall we treat the clusters? '*ticket*' and '*flight*' are also occurring as lone words, and they may share features with their clustered version. Moreover, considering those clusters as wholes, we would generate scarce, thus ill-modeled, entities, while depriving the lone words from statistical confirmation. So we'll read all the word component of a cluster. and consider only the head word of the cluster when looking for dependencies in through the tree.

a.3) tree structure

• *visualization*

```
                S
 ------------------------------------
 NP                     VP
 ---     ----------------------------------------------
  *   show     NP                    NP
               ---     ---------------------------------------------------
               me   NP               PP                 ADJP
                    ---        ------------------   ---------------------
               all the flights    PP          PP        early        PP
                            -------------  -----------        -------------
                            from    NP   to      NP            in        NP
                                    ---          ---                     ---
                                  Dallas       Denver              the morning
```

tree development of a sentence

figure 3.1.2

• *definition*

the tree is made of;

- nodes = tags
- leaves = words entities
- phrases = syntactic branches, with nodes and leaves.

The structure is a recursion of branches; a branch is called either a proposition, or a phrase.

• *building*

There are two main steps to get to the tree;

- assigning tags to the text
- bracketing the tagged text

they won't be exposed here in detail, cf. [15], but the dominant feature is the succession of *automatic parsing* and *human corrections..*

Though this project is young and bound for further studies and improvements, there is little hope to achieve an entirely automatic parsing. This is a point to ponder on if needing it directly for speech recognition .

• *proprieties*

The tree offers a multi-dimensional structure.

-> different navigations

Apart from the linear reading of words, there are all kind of recursive navigation on the branches i.e. phrases.

-> enable distance free relations

For example, jump between brother phrases, or from a parent phrase toward its ramifications. This is exactly the propriety we need for our LDDs.

3.2.b) Formalism

To define LDDs, we're using formal relations in the tree, let us define briefly the vocabulary and tools involved.

b.1) objects

• *phrase*

This is a proposition, i.e. a branch of the tree. It is either a simple word, or a complex phrase, in the latter case the first element is a tag giving rise to a sub-phrase.

• *tag-entity*

This is one of the 14 different labels of the tagset. A tag-entity in a phrase entails a node in the tree. Yet it is basically treated the same way as a word-entity, i.e. an instanciate of the tree structure.

• *word-entity*

It designates any leaf of the tree, including regular words, word clusters, null elements. A word entity is a ultimate - or degenerated- phrase.

• *word*

(regular)

This stands only for items of the vocabulary - typically, the dictionary - , it excludes clusters, but we decided to keep null elements. Besides, some words are replaced by a class, such as <place>, <day>, <month>, <number>, because of the particularity of ATIS. This is developed in *chapter 4.3* on sparseness. Classes are enumerated in program *index.h*, annex III.

b.2) relations

• *tree-parent*

Applies to phrases, a phrase A begets a phrase B if A contains B. By vocabulary abuse, we apply this to entities, either tag or word. entity A is tree-parent of entity B if A is father of B in the tree. An entity has one, and one only, tree-parent, except for the root of the tree. Note that word-entities are never tree-parents.

• *tree-brother*

Two phrases are brother if they have the same parent. idem for entities. An entity may have one brother or several or none.

• *ldd-brother*

(also called *brother*)

Basically, it applies to words. It can be define on word entities and then use the function "Head-Word" yields secular words. More than that it will be generally defined on phrases, thus including the word-entity case.

• *ldd-parent*

(also called *parent*)

we apply the same rule as above.

b.3) functions

• *Parent-Tag (entity) = tag*

The tag which gave rise to the word-entity or tag-entity, i.e. which is at the preceding node.

• *Head-Word (word-entity) = word*

On a regular word it's identity. On a words cluster, it is the head of the cluster. We will simply define it as the last word of the cluster. Yet this definition is somewhat crude. Moreover it should be highly dependent of the tongue. In our English data though, it works pretty well.

3.2.c) Brother connection

c.1) intuition

the idea of relating entities on the same level presents ; parallelism, symmetry, repetition. These criteria are obvious in the tree, common to perceive, maybe common in thinking and in speaking?

c.2) definition

Before settling on a final version a first definition was explored.

• *first definition*

entities are brothers if their parent-tags are identical and brothers in the tree

A ldd-brother1 *B*<=> *parent-tag(A)* tree-brother *parent-tag(B)* *parent-tag(A) = parent-tag(B)*

Definition I

((...	(<u>PP</u>	**from**
		(NP Dallas))
	(<u>PP</u>	**to**
		(NP Denver)))
...)		

'from' --*Brother*--> 'to'

We look for the relation between (A,B) in Im(Head-Word), i.e. in case of a cluster we consider its head word only.

• *limitations*

- It ignores brotherhood across propositions, for example :

((...	(PP	from
		(<u>NP</u> Dallas))
	(<u>PP</u>	to
		(<u>NP</u> Denver)))
...)		

'Dallas' --NOT *Brother*--> 'Denver'

- It distinguish the structural roles of the objects *word* and *proposition* which goes against the tree recursive spirit.

• *generalization*

We derive naturally an extended relation from the recursion of the tree, replacing "word-entity" by "phrase". And this comes down to introducing relation between parent-tags; **entities are general *brothers* if their parent-tags are *brothers*** , according to first definition or according to generalization , the definition is recursive, just as the tree is.

A *ldd-brother* B <=> parent-tag(A) *ldd-brother* parent-tag(B)

Definition II

this time the relation may be applied to tags too

```
((...  (PP   from
             (NP Dallas))
       (PP   to
             (NP Denver)))
...)
```

'NP' --g^{al} *Brother*--> 'NP'

'Dallas' --g^{al} *Brother*--> 'Denver'

It accepts "cousin" words like 'Dallas' - 'Denver' thus longer dependencies.

```
                 S
------------------------------------
 NP                        VP
 ---  ---------------------------------------------------
  *   show   NP                        NP
             ---   ---------------------------------------------------------
             me     NP                 PP                      ADJP
                    ---        --------------------     ---------------------
              all the flights    PP            PP       early          PP
                           -------------  ----------               -------------
                           from    NP     to     NP                 in        NP
                                   ---           ---                          ---
                                 Dallas        Denver                    the morning
```

Does it jeopardize consistency? Not with the strong constraint on Tags identity, which prevents 'Dallas' and 'Denver' to be brothers of 'the morning'.

c.3) note

• *transitivity potential*

Notion of brotherhood naturally suggests transitivity ; brothers of brothers are brothers. This fact is verified if several words occur as brothers in the same phrase. So we are tempted to "merge" brotherhood. This could be a mean to deal with sparseness ; arguing that the resulting unseen brothers deserved to occur, had data been only larger. Yet it would raise some statistical problems, as to what probability they should be granted, and it might generate misleading if unseen brothers outnumber seen brothers. So, if this idea is applied it need some refinement first.

• *associativity potential*

How should the model deal with successions of brothers in a phrase? The relation is so selective, it would be a waste not to use all found brothers. Yet is the brother n-gram relevant? taking coocurrences in account will probably result statistically in sparse data and unreliable estimates. Just as transitivity, associativity potential could be developed, under certain conditions, but first of all, let us evaluate the worth of simple brotherhood dependency.

3.2.d) Parent connection

d.1) intuition

- looking for contents dependencies and word associations ; for example, 'flight' may induce 'book' , 'book' may induce 'ticket', 'from' may induce a place name.
- suspecting importance of the node ; the word occurring at a node may be the head of the coming branches / phrases, for example, in figure 3.1.1 and 3.1.2, 'the nonstop flight' would be head of the phrases 'from Dallas to Denver' and 'early in the morning'.
- The second natural tree relation; fatherhood

d.2) definition

A ldd-parent1 *B* <=> *parent-tag(A)* tree-parent *parent-tag(B)*

Definition I

```
(PP    from
       (NP    Dallas))
```

'from' --*Parent*--> 'Dallas'

as we did for the brothers, we can draw a generalization :

A ldd-parent *B* <=> *parent-tag(A)* ldd-parent *parent-tag(B)*

Definition II

yet this might not be as clear as in the brother case;

- a jump is already performed by definition
- applying the generalization add all the elder ldd-parents to the list of ldd-parents.

The adopted view is to keep the closest parent.

```
                  S
-----------------------------------
NP                        VP
---    ----------------------------------------------
 *    show   NP                       NP
             ---    ---------------------------------------------------
             me     NP                PP                ADJP
                    ---         -------------------   ---------------------
             all the flights    PP           PP       early         PP
                             -------------  ----------            -------------
                             from   NP   to    NP                 in        NP
                                    ---        ---                          ---
                                  Dallas     Denver                   the morning
```

sons of 'show'

d.3) note

• *tree point of view:*

this relation is at the root of the tree structure: it should inherit the consistency of creating a node there.

• *linguistic point of view:*

the parent is assumed the head of the following proposition. It is natural enough in English, it may need adjustment according to the tongue (German...)

• *transitivity*

In a different way as brothers, parents have also a transitive quality ;

'*arriving*' induces '*after*'

'*after*' induces '*o'clock*'.

It is verified that 'arriving' induces '*o'clock*'.

Now using that kind of propriety requires some caution, as previously explained.

3.3 First glance : Mutual Information

3.3.a) Collecting : tables and vectors

a.1) table of dependency

A table for a given dependency lists the following data; For each item of the Vocabulary (regular word), a list of the items which depends on him, says the training data. For each couple thus listed we store its occurrence count.

'from'			
	-->	'to'	371
	-->	'on'	4
	-->	'into'	3
	-->	'back'	2
	-->	'after'	1
	-->	'at'	1
	-->	'from'	1
'leaving'			
	-->	'arriving'	20
	-->	'returning'	9
	-->	'departing'	1
	-->	'going'	1

someentries in the Brother table

figure 3.3.1

'after'			
	-->	<mixed number>	121
	-->	'o'clock'	35
	-->	'day'	4
'arriving'			
	-->	'at'	13
	-->	'before'	9
	-->	'approximately'	3
	-->	'into'	1

someentries in the Parent table

figure 3.3.2

a.2) dependency vector

A vector is defined as follows;

FORMAL	TRAINING	RECOGNITION
• *word*	*'to'*	*<?>*
past information:	|	|
• *Bigram*	*<PLACE>*	*<PLACE>*
• *LDD I (Brother)*	*'from'*	*'from'*
• *LDD II (Parent)*	*'flights'*	*'flights'*

dependency vectors

figure 3.3.3

The past information stored in the vector is the source that will help us uncovering outputs. The behavior of this backing source is deduced from the tables described in *a.1)* .

3.3.b) Consistency : mutual information

assessing their adequation to recognition?

Before going any further in the recognition process let us have a look at the collected tables of brothers and parents. As seen previously, mutual information gives us some clue on the information stored in a dependency. It can tell whether the couple covers an actual correlation in the source's outputs or is rather an accident, if not a misleading track.
for example, let us look at some brothers of 'from' :

Brother			Mutual Count	Mutual Info $.10^{-5}$
'from'				
	-->	'to'	371	10 000
	-->	'on'	4	- 40
	-->	'into'	3	60
	-->	'back'	2	40
	-->	'after'	1	- 2
	-->	'at'	1	- 8
	-->	'from'	1	- 20

mutual info of 'from' as a brother

figure 3.3.4

The higher the probabilities of words, the higher the risk to be mislead by accidental occurrences. On the whole in this case, mutual informations add to a positive value. 'from', as an output of the "brother source" succeeds in uncovering information about the source to recognize.

b.1) Brothers

• *sparseness*

Compared with the amount of data, there are few brother occurrences. A sensible amount of occurrences are isolated (singletons). Yet observing them, most make sense from a simple "interpretative" point of view.

• *function words*

Some trends can be observed. Mainly, links between "function words" are common ;

'from'-->'to'
'before'-->'after'
'on'-->'on'

more semantic information appears sometimes,

'leaving'-->'arriving'
<place> --> <place>

But not so commonly. This is due to the low complexity of the sentences, which can be accounted for by their limited length.

b.2) Parents

• *trash occurrences*

A lot of couples occur only once. If high probability words are involved, then we can draw a conclusion, they have little chance to be parent and sons, but if low probability words are involved, we cannot really conclude, resulting Mutual Information may be positive, even high, yet who can tell if the dependency is reliable?
Besides, conjunction words like *'and'*, *'then'*, end up with so numerous sons, that it doesn't express much information , typically, mutual information will be negative for such words.

Now the table reckons a lot of couples, much more than the brothers table, and apart from the trash occurrences, still a lot of couples capture a positive mutual information.

• *solve the contiguity problem*

Particular cases of parents are bigrams, but the main feature of those dependencies is their ability to jump over contiguity ; even if they only dealt with very local relations, they could be a good improvement to bigrams. We find such dependencies as

'book'-->'flight', 'book'-->'seat'
'show-->'list'
'arriving'-->'O'clock'
'flights'-->'from', 'flights'-->'to'

So, it is time to use those dependencies effectively and see how much they will help recognition.

4. The Recognition Task

definition-optimization

4.1 Context & Hypothesis

4.1.a) Structure : ideal case

a.1) from text

We are staying apart from acoustic domain, as exposed in *§ 2.1* the language term is our sole preoccupation. therefore we conduct our recognition experiment on text. We will recognize "regular" words, in the sense defined in *§ 3.2*, i.e. an isolated word or one of the few generic classes.

a.2) known previous words

The guess is made with the help of preceding words. In real case, there is no way to check if previous guesses were right. But to evaluate the potential worth of this source, we are using the correct preceding words.

a.3) known skeleton

The joint sources we're using to guess outputs from text source, are the bigram source, the brother source, the parent source. Which means that for each output we supposedly know its bigram, its brother and its parent. Now, to have such elements as brother and parent we need to know the tree structure of the string we are guessing.
We are actually uncovering word component of a blind syntactic tree, in other words, we know the skeleton of the sentences.

4.1.b) Data : Atis or WSJ ?

b.1) alternative : Wall Street Journal

up to now, we quoted results on ATIS, but we dispose of another corpus, parsed the same way as ATIS, the WSJ corpus; abstracts from the Wall Street Journal.
Here is a parsed sentence from WSJ :

```
(
   (S  (NP   (NP      Pierre Vinken)
       ,
       (NP   (NP      61 years)
             (ADJP    old))
       ,)
   will
       (VP   join
             (NP      the board)
             (PP      as
                      (NP     a non executive director))
             (ADVP (NP        Nov. 29))))
.)
```

parsed sentence from WSJ

figure 4.1.1

b.2) advantages and drawbacks

• *scale*

WSJ corpus is way wider than ATIS corpus. There are 1,300 parsed sentences in ATIS, and more than 10,000 in WSJ. This is a major point dealing with statistics, where sparseness and unreliability are no slight worries.

• *vocabulary*

On the other hand, ATIS' vocabulary is much more reasonable than WSJ's one. On a 1,000 sentences base, ATIS has a vocabulary of ~300 words and WSJ of ~5,000 words, and this figure can still increase much on total WSJ data. ATIS context is very limited : questions on flights and travels in US, compared to the scope of financial articles.

• *structure*

Now, sentences of WSJ tends to be much longer and more complex than usual interrogations found in ATIS. This accounts in part for the vocabulary increase, mainly it stands as an asset concerning the search of parents and brothers through the tree.

b.3) choice criterion ; mutual information

Ideally, we would have the ATIS limited vocabulary with the WSJ amount of data and syntactic wealth. As this is but a dream, we have to compromise. To settle a choice, we looked at dependency tables and mutual information figures. Too many singleton and non consistent dependencies were occurring with WSJ. We will run our evalution on ATIS.

4.1.c) Task : prediction

c.1) without acoustic evidence

Since we are using language model only, our evaluation is on the potential predictions of next words, i.e. the potential loss of perplexity.

c.2) from several predictors

We are using different information sources :

- Bigram
- Brother
- Parent

Each involves a model of ATIS source, we will unify the model to use it a coherent information source.

4.2 Processing -> stochastic / knowledge

4.2.a) Problematic

Here is some description of the way to collect, store and use the dependencies we have defined, on both knowledge based data and stochastic data, on both linear and multidimensional data.

4.2.b) Architecture:

b.1) source

[text]

->parsing->

[parsed text]

-> dependencies detection->

[vectorial text]

In further operations we will use vectorial text:
words are replaced by Dependencies Vectors.

b.2) training set

[vectorial text]

->parameter estimation->

[tables of dependencies]

b.3) adapting set

[vectorial text]

->factor converging->

[adapted factors]

b.4) testing set

[vectorial text]

->recognition (evaluation) ->

[word probabilities]

4.2.c) Proprieties

c.1) modular attempt

This architecture distinguish

- *information used to track dependencies*

-> operations on an information rich corpus; the parsed text.

- *information used to recognize outputs*

-> operation on essential data; the dependency vectors

c.2) limitation

Yet we must not ignore that the gap is somewhat artificial; as long as we're using the corpus to determine the dependencies hence the vectors, it is part of every step.

4.3 Sparseness problem -> smoothing

Sparseness of data is an inherent propriety of any real text, and it is a problem that one always encounters while collecting frequency statistics on words and word sequences (N-grams, long distance couples...) from a text of finite size. This means that even for a very large data collection, the maximum likelihood estimation method (MLE) does not allow us to estimate probabilities of rare but nevertheless possible word sequences-many sequences occur just once (singletons), many more do not occur at all.

For unseen sequences maximum likelihood estimator yields a null probability, which entails infinite uncertainty. It is not acceptable for a recognizer. Moreover, for limited data collections, not only are unseen sequences more numerous, but a lot of sequences will occur as singletons; occurrences are more loosely related to the actual distribution of the language source; estimating the probability of sequences requires another statistic than MLE.

The ATIS corpus we're using raises acute sparseness problem, we expose them and some remedies.

4.3.a) Sparseness; the figures

a.1) Context

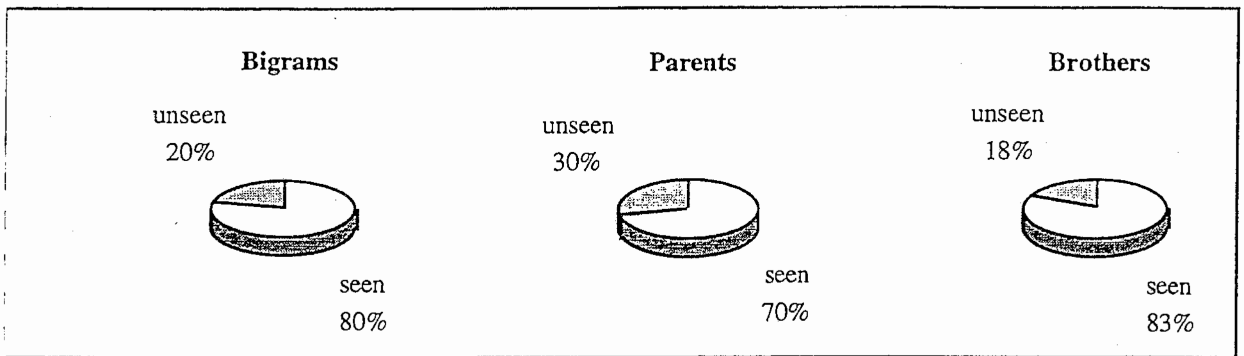
The file consists in 1382 sentences, of 16,273 words. We chose the following partition - we'll discuss that choice further -,

- 3/4 of the file, i.e. 1151 sentences, are for training purpose
- 1/4 of the file, i.e. 231 sentences, serves the testing.

In the training set, sequences of words (Bigrams and LDDs) are reckoned, those are the sequences that allow prediction in target texts, assuming it has the training text behavior. Now, two questions arise: do *target* sequences behave as their *training* homonyms? and more confusing, does the training homonym exist? In the testing set, we reckoned the actual sequences, to compare with the training sequences.

a.2) Figures

For each target sequence, we asked ; was this sequence seen in the training ?
We apply the poll to Bigrams, to Parents and to Brother.



unseen proportion for bigrams, parents, brothers

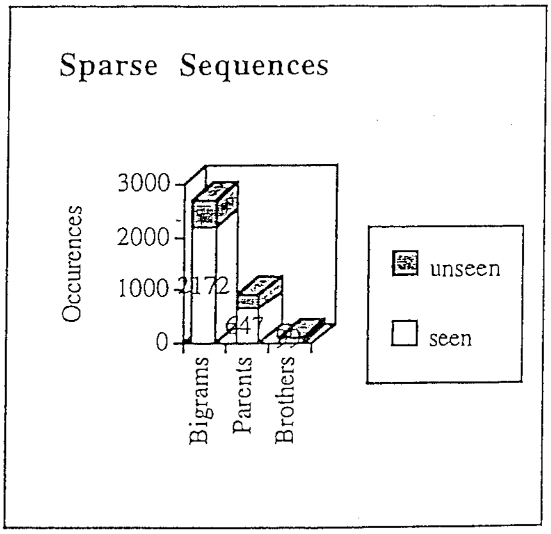
figure 4.3.1

For Bigrams, as for Brothers, 1 sequence out of 5 was not encountered in the training set. For Parents it's even worse, 1 sequence out of 3; it's obviously unrealistic to classify those unseen Parent-Son sequences as irrelevant compared to the seen ones, Something has to be done for those lively "phantoms".

Furthermore, as shown by *figure 4.3.2* , Brothers and Parents appear to be quite scarce,

- compared to Bigrams

Whereas any word can be associated with a bigram, even first words which are associated with the <start> state, only few seem to be related to a parent and much fewer to a brother, this stands for a first limitation of LDDs, but it is the price to pay for selective model, and it shall be kept in mind that LDDs are alone in their category, and deal with quite different information as bigrams.



unseen occurrences of bigrams, parents, brothers

figure 4.3.2

- in absolute

Regarding statistical need for consistency, Results aren't so reliable dealing with as scarce occurrences as brothers, even if the figures are faithful to the intrinsic quality of the Brother model. On top of that it can be suspected that sparse data artificially emphasize scarceness of selective models. So, we're not experimenting in the best conditions, but it is a first approach, it is interesting to see what it can tell.

4.3.b) Structural remedies

word classes

Classes of words are defined, so that probabilities are computed not on words but on their classes. Besides lowering the task complexity, they act as generalization factor and consistency accelerators, provided that they are well chosen. For example, correlation between '*from*' and '*New York*' may be the same as between '*from*' and '*Dallas*', therefore the correlation will be more effectively taken in account statistically, if seen between '*from*' and the class '<*place*>', it is all the more true than the data is sparse.

That kind of clustering can be done in a unsupervised way, by statistical iterations[1]

now, thanks to particularities of ATIS, some classes are very naturally defined, such as *place*, *day*, *month*, *number* ..., whose elements, we can assume, play the same role.

This clustering was performed even before getting to the results shown above.

4.3.c) Essential remedy : smoothing

c.1) smoothed estimation

Facing sparseness and inadequacy of MLE, an other statistic is applied. The main idea is to reduce unreliable probability estimates given by the observed frequencies and redistribute the "freed" probability "mass" among sequences which never occurred in the text. The redistribution may be uniform - affect the same probability to all unseen sequences- or follow some criterion according to the unseen sequence.
As a result recognition of the training set is not so good as performed with the MLE, which is the best estimator, but applied on a test text it gives better results; it helps going beyond the limitations of a training set.

c.2) Katz's estimate

We applied a smoothed estimate for bigrams as proposed by Slava M. Katz. The reduction of unreliable probability is achieved by Turing's like estimates.
Let N be a sample text size and let n_r be the number of words (m-grams) which occurred in the text exactly r times, so that

$$N = \sum_r r n_r \tag{4.3.1}$$

Turing's estimate P_T for a probability of a word (m-gram) which occurred in the sample r times is

$$P_T = \frac{r^*}{N} \tag{4.3.2}$$

where

$$r^* = (r+1)\frac{n_{r+1}}{n_r}. \tag{4.3.3}$$

A procedure of replacing a count r with a modified count r' is defined as "discounting" and a ratio r'/r as a discount coefficient d_r. When $r' = r^*$ we have Turing's discounting. An m-gram $w_1 .. w_m$ is denoted as w_1^m and the number of times it occurred as $c(w_1^m)$. Then the Turing estimate is

$$P_T = \frac{c^*(w_1^m)}{N} \tag{4.3.3'}$$

where

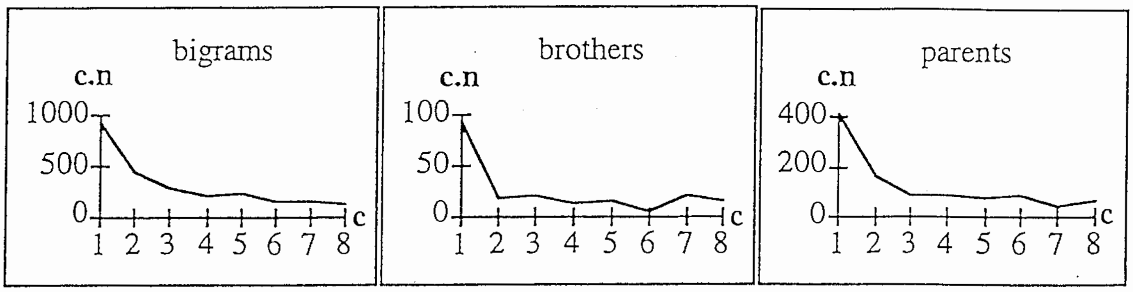
$$c^*(x) = (c(x)+1)\frac{n_{c(x)+1}}{n_{c(x)}}. \tag{4.3.2'}$$

How can we interpret this replacing?

The discount coefficient is $$d^* = \frac{(c+1)n_{c+1}}{cn_c}$$

i.e. $$d^* = \frac{\text{contribution(classe } c+1)}{\text{contribution(classe } c)}$$

It smoothes the hectic repartitions between different classes, most of all, the edge effect of singletons, *figure 4.3.4.* displays the repartitions of classes for bigrams, brothers and parents relations. The smoothing forces $c^*n = (c+1)(n+1)$



classes contributions

figure 4.3.4

It follows that the total probability estimate for the set of sequences that actually occurred in the sample is

$$\sum_{w_1^m : c(w_1^m) > 0} P_T(w_1^m) = 1 - \frac{n_1}{N}. \tag{4.3.4}$$

This in turn, leads to the estimate for the probability of observing some previously unseen m-gram as a fraction n_1/N of singletons in the text:

$$\sum_{w_1^m : c(w_1^m) = 0} P_T(w_1^m) = \frac{n_1}{N}. \tag{4.3.5}$$

δ_c is defined as;

$$\delta_{c(w_1^m)} = P_{ML}(w_1^m) - P_T(w_1^m) \tag{4.3.6}$$

Where P_{ML} is the maximum likelihood estimate. As maximum likelihood estimates sum up to 1 on the sample's sequences, and given the previous sigmas ;

$$\sum_{w_1^m : c(w_1^m) > 0} \delta_{c(w_1^m)} = \frac{n_1}{N}. \tag{4.3.7}$$

$\delta_{c(w_1^m)}$ can be interpreted as the contribution of an m-gram w_1^m with a count $c(w_1^m)$ to the probability of "unseen" m-grams. Explicitly ;

$$\delta_c = \frac{c}{N} - \frac{c^*}{N} = (1 - d_c)\frac{c}{N}. \tag{4.3.8}$$

An analogous contribution is defined for conditional probabilities

$$\delta^{cond}_{c(w_1^m)} = \left(1 - d_{c(w_1^m)}\right)\frac{c(w_1^m)}{c(w_1^{m-1})}. \tag{4.3.9}$$

An estimate is derived from $\delta^{cond}_{c(w_1^m)}$ The sum of all contributions is then distributed on the unseen m-grams proportionally with the estimate of the *(m-1°*-gram, so that the definition is recursive. Let's focus our interest toward the bigram case (or the LDD case, both involving pairs of words), for existing sequence the estimate is

$$P_s(w_2|w_1) = d_{c(w_1,w_2)}\frac{c(w_1,w_2)}{c(w_1)}. \tag{4.3.10}$$

ß is defined as the sum of the contributions of existing bigrams starting with w_1 ;

$$\beta(w_1) = \sum_{w_2:c(w_1,w_2)>0} \delta^{cond}_{c(w_1,w_2)} = 1 - \sum_{w_2:c(w_1,w_2)>0} P_s(w_2|w_1) \tag{4.3.11}$$

This gives an estimate of the sum of conditional probabilities of all words w_2 which never followed w_1 .ß is distributed among unseen w_2 according to their probability estimate

$$P_s(w_2|w_1) = \alpha P_s(w_2) \tag{4.3.12}$$

where α is a normalizing constant,

$$\alpha(w_1) = \frac{\beta(w_1)}{\sum_{w_2:c(w_1,w_2)=0} P_s(w_2)} = \frac{1 - \sum_{w_2:c(w_1,w_2)>0} P_s(w_2|w_1)}{1 - \sum_{w_2:c(w_1,w_2)>0} P_s(w_2)} \tag{4.3.13}$$

c.3) resolving edges' problems

• *superior edge*

A modified version is proposed which doesn't discount high values of count c>k, considering them as reliable, yet leaves intact the estimate n_1/N for the probability of all unseen m-grams. The coefficient d_r of this new discounting is calculated to abide by those constraints,

$$\text{for } r > k\,,\ d_r = 1$$

$$\text{for } 1 \le r \le k\,,\ d_r = \frac{\dfrac{r^*}{r} - \dfrac{(k+1)n_{k+1}}{1.n_1}}{1 - \dfrac{(k+1)n_{k+1}}{1.n_1}}. \tag{4.3.14}$$

In fact, newly defined contributions $(r/N - r'/N)$ are proportional to the Turing's contributions $(r/N - r^*/N)$

As for the value of the parameter k, k=5 is recommended, though the model is not very sensitive to that.

• *inferior edge*

Now, it might very well happen that for several r class *r(w)* is inexistent, i.e. no sequence occurs exactly r times in the sample text. To prevent ill disappearances and transferees, we added the following rule;

$$\begin{cases} r^* \neq 0, r' = r^* = \dfrac{(r+1)n_{r+1}}{n_r} \\ r^* = 0, r' = \dfrac{(r+1+p)n_{r+1+p}}{n_r}, p{:}\, p > 1, \begin{cases} n_{r+1+p} \neq 0 \\ n_{r+p} = 0 \end{cases} \end{cases} \qquad (4.3.15)$$

This doesn't affect the preceding result as, owing to (4.3.2)

$$r^* = 0 \Leftrightarrow n_{(r+1)} = 0$$

and

$$N = \sum_r rn_r = \sum_{r:n_r>0} rn_r \qquad (4.3.1)$$

For example, the first sum formerly calculated (4.3.4) is now

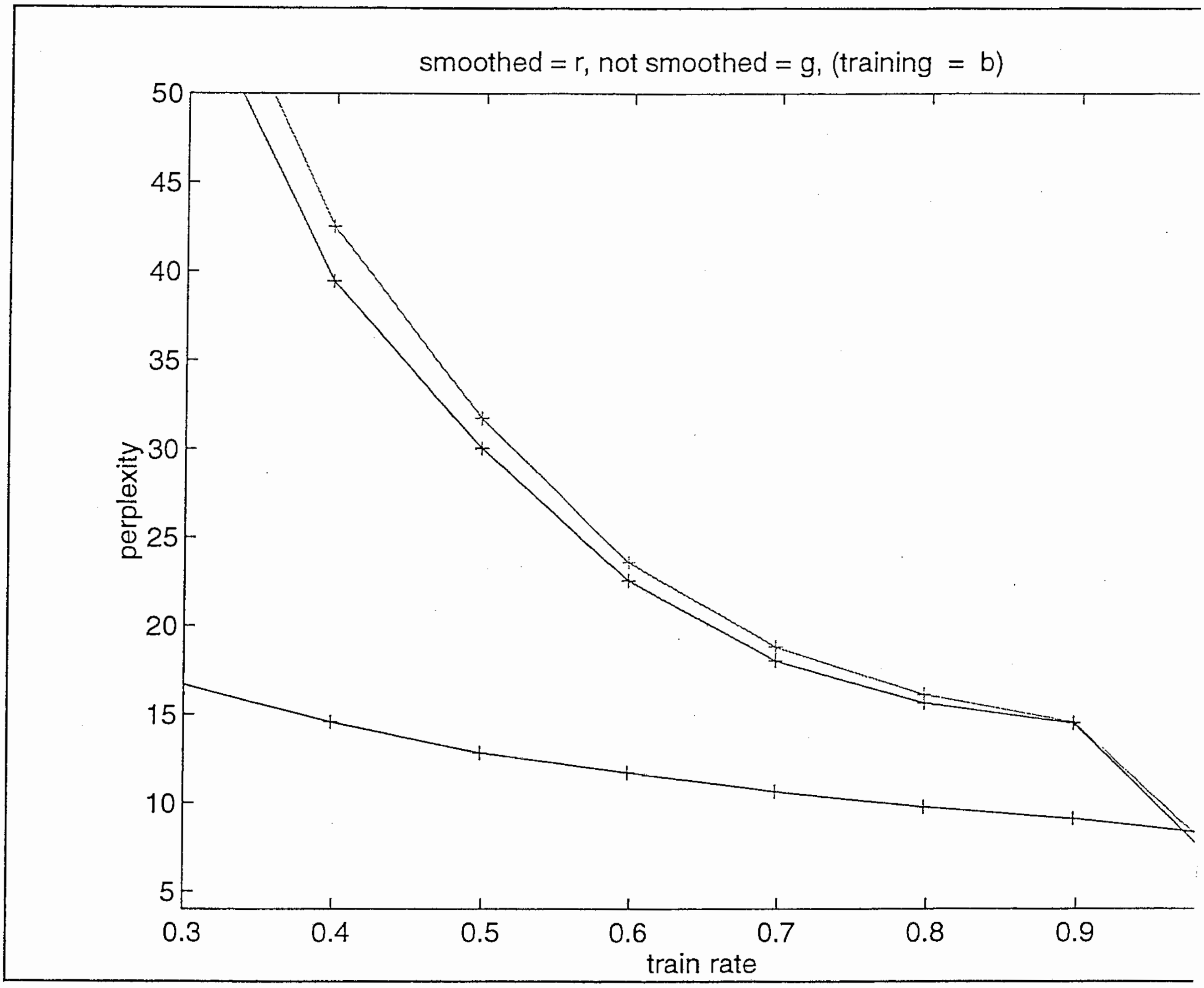
$$\begin{aligned} \sum_{w:r(w)>0} P'(w_1^m) &= \sum_{r:r>0,n_r>0} n_r \frac{r'}{N} \\ &= \frac{1}{N} \sum_{r_i} n_{r_i} \frac{r_{i+1} n_{r_{i+1}}}{n_{r_i}} \\ &= 1 - \frac{n_{r_1}}{N} \end{aligned} \qquad (4.3.4')$$

which is the same as (4.3.4), provided that there are singletons, i.e. $r_1 = 1$ Our rule is a mere re-ordering of the classes but keeps everything else even, it makes sure that counts' sliding are operated only between existent classes. It makes sense since classes set in the sample text englobe any further word or sequence ; if the sequence occurred in the sample text, it belongs to its occurrence's class, if it didn't occur, it belongs to the "occurrence 0" class, i.e. the class of unseen sequences .

c.4) what to smooth ?

The method was originally made up for bigrams. Adapted and efficient, it settles the problem of unseen bigrams. What about brothers and parents? As they are combined with bigrams, the necessity of smoothing is no longer so urgent; if they yield null result, the bigram's conditional probability will just take over.

Yet null probability is always a severe loss, moreover, the sparseness figures (cf. former § *figure 4.3.1*) shows LDDs suffer from unseen occurrences as much as bigrams. Therefore it is logical to think ML estimate is no more adequate for them as it was for bigrams. The graph in *figure 4.3.3* compares perplexities of the testing set, whether LDDs are smoothed or not.



smoothing LDDs

figure 4.3.3

Though slightly better, the smoothing of LDDs is not so helpful as could be expected, several reasons may account for it:

- a scale effect

Bigrams are much more numerous and used than brother or parent, who are "optional", so that some changes on the statistics of the latter don't achieve much improvement, this point will be discussed in more detail later.

- an unadapted smoothing

Now, "ill conditioned" doesn't mean the statistics of LDDs are inconsistent, neither does it find their sparseness desperate, it simply suggests this smoothing is not adequate. Specific smoothing for LDDs could be explore, either purely statistic, or somewhat structural, as deriving new brotherhood transitively from existent ones.

4.4 Correlation question-> combining

4.4.a) Back to history

A word is granted a history vector by the model,

$$\tilde{h} = \left[w_{big}, w_{bro}, w_{par}\right] \tag{4.4.1}$$

Probability given history (cf. § 2.1) is expressed according to the model as

$$\begin{aligned} P\left(w_i \middle| w_{i-1},..,w_1\right) &= P\left(w_i \middle| \tilde{h}_i\right) \\ &= P\left(w_i \middle| w_{i-1}, w_{bro_i}, w_{par_i}\right) \\ &= f\left(w_i, w_{i-1}, w_{bro_i}, w_{par_i}\right) \end{aligned} \tag{4.4.2}$$

What is the explicit form of f?

Calculating f exactly, entails knowing the correlation between all pairs of elements in the history vector. Getting those correlations statistically, using crude samples' counts, is out of the question; introduction of parameters in this non-parametric question is not justified here. cross counts will be too scarce. On the other hand, Getting them mathematically requires some circumvolutions and additional hypothesis;basically our hypothesis is that strong correlations exist between $w_i \leftrightarrow \tilde{h}_i$

nothing was assumed concerning the cross correlations in $\tilde{h}_i$.

It is not necessary to look for an exact calculation of f , we can reach a satisfying estimate not bothering explicit correlations; by linear interpolation and stochastic converging

4.4.b) Linear combining

b.1) expression

We have estimates of each conditional probabilities, we want to combine them linearly. A general expression is,

$$P\left(w \middle| w_{big}, w_{bro}, w_{par}\right) = \alpha_1 P\left(w \middle| w_{big}\right) + \alpha_2 P\left(w \middle| w_{bro}\right) + \alpha_3 P\left(w \middle| w_{par}\right) \tag{4.4.3}$$

$$\alpha_1 + \alpha_2 + \alpha_3 = 1$$

That way the estimate keeps evidently the statistic propriety to sum up to 1.

b.2) application ; context cases

We will handle several cases, and apply the following distributions:

bigram	brother	parent	$\rightarrow$	distribution $P(w\|\tilde{h})$
×			(1)	$P(w\|w_{big})$
×	×		(2)	$(1-\lambda_1)P(w\|w_{big})+\lambda_1 P(w\|w_{bro})$
×		×	(3)	$(1-\lambda_2)P(w\|w_{big})+\lambda_2 P(w\|w_{par})$
×	×	×	(4)	$(1-\lambda_4-\lambda_3)P(w\|w_{big})+\lambda_3 P(w\|w_{bro})+\lambda_4 P(w\|w_{par})$

It can be assumed that some proprieties of cases (2) and (3) are held in case (4). Intuitively, if Brother Model and Parent Model are orthogonal, their weights relatively to Bigram Model should be blind to case (4);

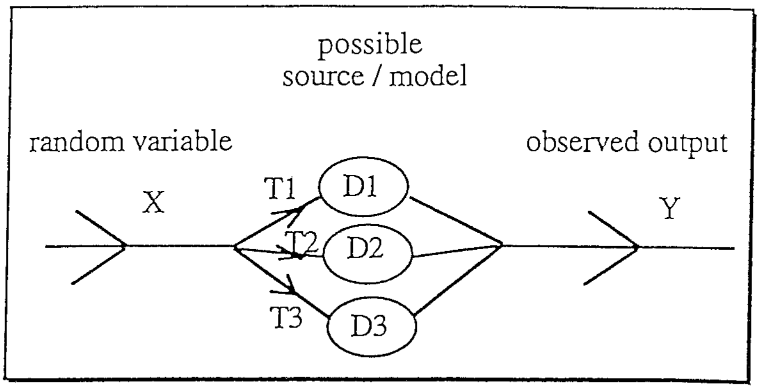
$$\begin{cases} \dfrac{\lambda_3}{1-\lambda_4-\lambda_3} = \dfrac{\lambda_1}{1-\lambda_1} \\ \dfrac{\lambda_4}{1-\lambda_4-\lambda_3} = \dfrac{\lambda_2}{1-\lambda_2} \end{cases} \tag{4.4.4}$$

$$\Leftrightarrow \quad \begin{cases} \lambda_3 = \dfrac{\lambda_1 - \lambda_1\lambda_2}{1-\lambda_1\lambda_2} \\ \lambda_4 = \dfrac{\lambda_2 - \lambda_1\lambda_2}{1-\lambda_1\lambda_2} \end{cases} \tag{4.4.4'}$$

If they are not orthogonal it can be pointed out that our formulas can't really take their correlation in account. In order to take advantage from their virtual correlation we should pick combined information from the training text and inject it in the distributions. This would magnify the sparseness problem and complicate excessively the probability estimates. In the end, computing λ_3, λ_4 is a compromised approach to Brother / Parent correlation, which allows us not to get burdened by more parameters.
Now how do we estimate the linear weights λ ?

b.2) interpretation

Linear combination of probabilities might appear unnatural -compared to products- yet this reflects an intuitive concept of *distributions mixture*[3].
Applying *figure 4.4.1* scheme to LDD context : three possible distributions $D_{big}, D_{bro}, D_{par}$ are chosen with respective probability $\alpha_1, \alpha_2, \alpha_3$, (standing for statistics T1, T2, T3) to spawn the observed word w (standing for observed output Y) .



combining models linearly

figure 4.4.1

4.4.c) EM algorithm

c.1) Intuitive aspect

Such a mixture emerges most naturally dealing with Gaussian statistics (red and white blood cells, speech and silence...), and the Gaussian case allows simple and elegant estimation as we'll expose next.

Generally speaking, let N_A and N_B be two distributions whose sets of parameters are Θ_A, Θ_B.they are chosen respectively with probability λ_A, λ_B, the parameter we will have to estimate is

$$\Theta = [\Theta_A, \Theta_B, \lambda_A, \lambda_B]$$

In our combining case, we don't have to compute Θ_A, Θ_B which are implicitly estimated from the training set, but we have to estimate λ_A, λ_B. - the EM algorithm is more powerful than needed, anyway let us expose it generally.

The probabilities of observations (y_i) are

$$f(y_i) = \lambda_A N_A(y_i) + \lambda_B N_B(y_i) \tag{4.4.5}$$

If it were known when A was used and when B was used in generating observations, then ML estimates could be used for λ_A, λ_B (multinomial) on the one hand, and for Θ_A, Θ_B on the other hand. Since that information is hidden, the solution is to assume some parameter Θ , compute that way when and how often each distribution is expected to be used given the observed data; the expected statistics help to compute new estimates of parameters. Then start with the new estimate and iterate.

The probability that A was used, given that y_i was observed is

$$P_\Theta(X_i = A | y_i) = \frac{P_\Theta(X_i = A, y_i)}{P_\Theta(X_i = A, y_i) + P_\Theta(X_i = B, y_i)} \tag{4.4.6}$$

given f formula (4.4.5),

$$P_\Theta\left(X_i = A \middle| y_i\right) = \frac{\lambda_A N_A(y_i)}{\lambda_A N_A(y_i) + \lambda_B N_B(y_i)}. \tag{4.4.7}$$

The expected number of times A and B were used are

$$T_{0A} = \sum_{i=1}^{n} P_\Theta\left(X_i = A \middle| y_i\right)$$
$$T_{0B} = \sum_{i=1}^{n} P_\Theta\left(X_i = B \middle| y_i\right)$$

T_{0A} is a "statistic of order 0".

Applying Maximum Likelihood estimate for multinomials,

$$\begin{aligned} \tilde{\lambda}_A &= \frac{T_{0A}}{T_{0A} + T_{0B}} = \frac{T_{0A}}{n} \\ \tilde{\lambda}_B &= \frac{T_{0B}}{T_{0A} + T_{0B}} = \frac{T_{0B}}{n} \end{aligned} \tag{4.4.8}$$

therefore

$$\tilde{\lambda}_A = \frac{1}{n} \sum_{i=1}^{n} \frac{\lambda_A N_A(y_i)}{\lambda_A N_A(y_i) + \lambda_B N_B(y_i)} \tag{4.4.9}$$

A similar approach can be applied on higher statistics (means, covariance, ..)

c.2) Theoretical aspect

It can be demonstrated that the algorithm "Expectation-Maximization", which consists in

1. Choosing an initial set of statistics Θ
2. Computing Expectation $E_\Theta\left(\log P_{\tilde{\Theta}}(x, y) \middle| y\right)$
3. Maximizing it for $\tilde{\Theta}$
4. Setting $\Theta = \tilde{\Theta}$
5. Going back to 2. until convergence is enough

converges toward $P_{\tilde{\Theta}}(y)$.

An interesting form is found for exponential distribution

$$f_\theta(y) = h(\theta) e^{T(y)\theta}$$

with parameter $\theta = [\theta_1, .., \theta_n]$

and function $T(y) = [T_1(y), .., T_n(y)]$.

The EM comes down to solving the equation:

$$E_{\tilde{\Theta}}\left[T_i(x, y)\right] = E_\Theta\left[T_i(x, y) \middle| y\right] \tag{4.4.10}$$

where x stands for the used distribution, y for the observation, and $\tilde{\Theta}$.for the new estimate.

c.3) Application to dependencies combining

Let's deal with a case where two dependencies are combined. For example Bigram Model and Brother Model.

$$P\left(w\middle|\tilde{h}\right) = \lambda P\left(w\middle|w_{bro}\right) + (1-\lambda)P\left(w\middle|w_{big}\right) \qquad (4.4.5')$$

The statistic we would like to have to estimate λ is the number of time the Brother Model was used,

$$T = \sum_i \delta(x_i, Bigram\_Model)$$

Since we can't count this statistic we will use its expected value (i.e. use EM algorithm)

1. Start with an initial guess of λ (e.g. 0.5)
2. Solve $$E_{\hat{\lambda}}[T_i(x,y)] = E_{\lambda}[T_i(x,y)|y] \qquad (4.4.10')$$
3. Reestimate $\lambda = \hat{\lambda}$
4. Go back to step 2., until convergence criteria is satisfied.

Solving equation of step 2. we find eventually,

$$\hat{\lambda} = \frac{1}{N}\sum_{i=1}^{N} P_{\lambda}\left(X_i = BroM\middle|y\right) \qquad (4.4.8')$$

as in (4.4.6), $$\hat{\lambda} = \frac{1}{N}\sum_{i=1}^{N}\frac{P_{BroM}(y_i)P(X_i = BroM)}{P_{BroM}(y_i)P(X_i = BroM) + P_{BigM}(y_i)P(X_i = BigM)}$$

so that $$\boxed{\hat{\lambda} = \frac{1}{N}\sum_{i=1}^{N}\frac{P\left(y_i\middle|bro_i\right)\lambda}{P\left(y_i\middle|bro_i\right)\lambda + P\left(y_i\middle|big_i\right)(1-\lambda)}} \qquad (4.4.9')$$

This result is exactly the intuitive one found formerly (4.4.9). λ can be seen as an optimal weight of the Brother Model, compared with the Bigram Model.

c.4) Entropy's tale

The EM algorithm acts as an Entropy minimizer; the Logprob decreases at each step, as expectation of Log(P) is maximized.

• *Weights as consistency measures*

From the above cited point of view, weights can be seen as adapted to Information extraction; if a model succeeds in capturing Information from the source, it will be granted a high weight, whereas an unadapted model is granted a very low one. Actually, *weight* measures the Information consistency of a model, as far as the target text is concerned.

• *Comparing to Dichotomy*

The above remark also suggests other convergence algorithms of, minimizing Entropy - or Perplexity - though any algorithm is not assured to converge stochastically toward λ.

Actually, it can be demonstrated[1][3] that Entropy is convex varying with λ. As a control and out of curiosity, we implemented a simple Dichotomy algorithm. Roughly, the minimal Entropy found by both algorithm are the same, yet not exactly so. Now values of *weights* can vary to some extent. Mostly EM appears ways much quicker for a given precision.

	Iterations		λ		Perplexity
	Brother	Parent	Brother	Parent	
Dichotomy	20	20	0,501	0,276	20,43
EM Algorithm	5	6	0,462	0,311	20,52

Dichotomy / EM

figure 4.4.2

Obviously, we should not seek high precision for weights; if Dichotomy is reliable, it seems we can't assess weights with a precision superior to 10%, This point is confirmed in other articles[1][2], as is the very slow variation of Perplexity with those parameters.

5. Results

5.1 Prediction Ability

The task affects probabilities, computed from the described model, to each word of the target text. They finger out the ability of the model to predict those correct words. Here are the result of the different model on a test sentence.

Outputs	Probability (%)			
START ->	Bigram	Brother	Parent	Combine
I	9	-	-	9
would	21	-	-	21
like	64	-	66	64
*	56	-	58	56
to	16	-	59	39
book	8	-	-	8
a	30	-	-	30
one-way	7	-	-	7
flight	23	-	-	23
from	7	-	13	10
<place>	73	-	-	73
to	25	71	12	43
<place>	54	-	55	54
on	4	-	27	17
month	19	-	20	19
<numberth>	56	-	15	33
number	31	18	16	21
END	32	-	-	32

prediction scores of models

figure 5.1.1

In this case, we can see that

- bigrams are very efficient, so that parents and brothers don't add much.
- parents achieve quite a good job though.

Now, those features are actual trends, and we will see them again through other results.

5.2 Linear Weights

5.2.a) the last weights

- λ_1 for brothers combined with bigrams
- λ_2 for parents combined with bigrams

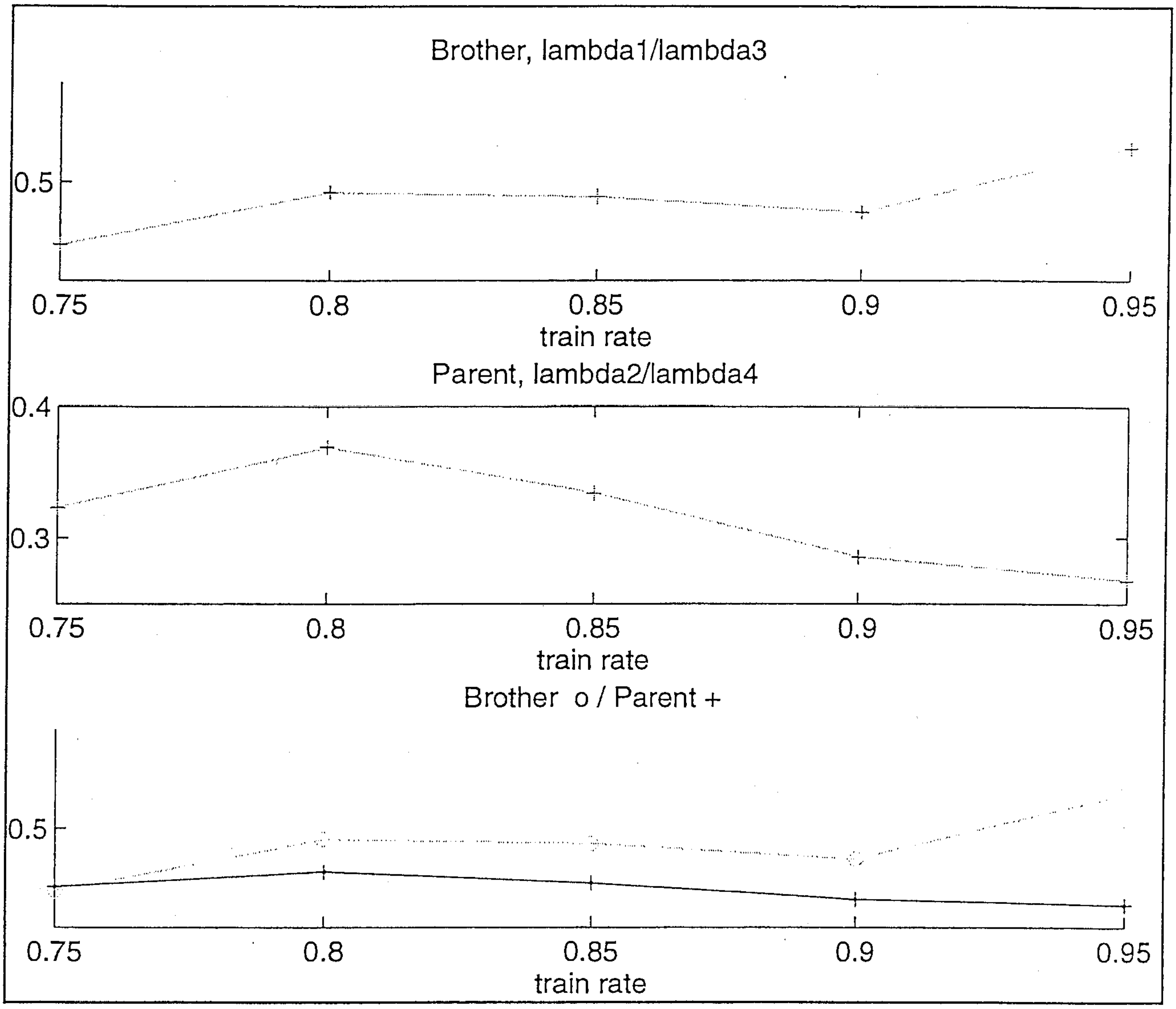
- λ_3, λ_4 for brothers and parents combined with bigrams

Now, the λ_3, λ_4 turn out to converge very closely to their estimates from λ_1 and λ_2 exposed in *§4.4*, using equation *(4.4.4')*. So that we don't need to compute them.

λ

5.2.b) Brothers vs. Parents

Here are the values of λ_1 and λ_2 for different amount of training.



Lambdas

figure 5.2.1

On the whole, it can be contended that λ_1~0.5

λ_2~0.3

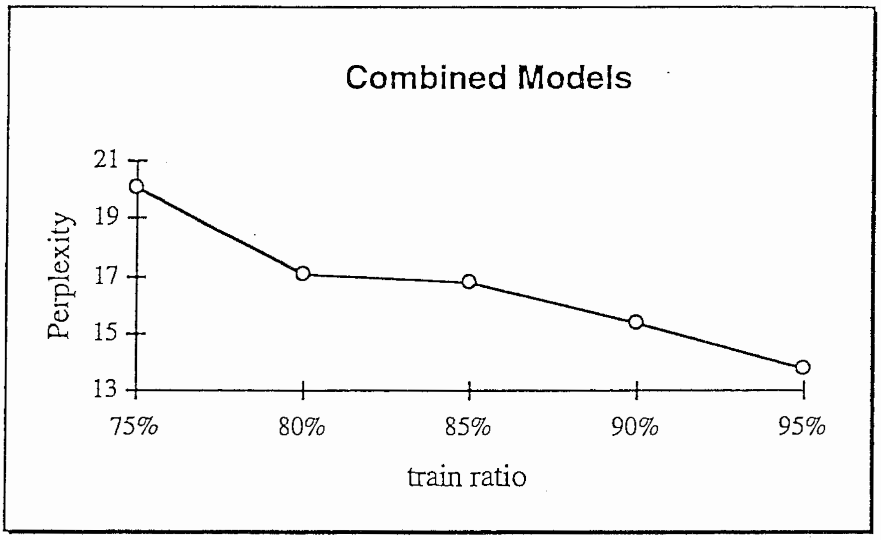
This means that brothers and parents are as consistent to give rise to the text as bigrams ; their distributions have very close probabilities to be chosen and foster the next output.

5.3 Perplexity

Perplexity is estimated according to the Logprob. The Logprob takes in account probabilities of correct words as shown above. The "Perplexity of the model" is the perplexity obtained on a test set. Yet it is interesting to cast a glance to perplexities computed on training and converging sets.

5.3.a) test Perplexity and amount of training

We divide the corpus into three sets ; training, converging, testing.
What amount of training should be used and what amount of testing ?
As we don't have a large corpus, using to much for training will yield a testing set reduced to irrelevancy.



test perplexity / training amount

figure 5.3.1

The dramatic decrease after 85% / 90% is due less to good training estimates than to unreliable testing estimates.

The observed perplexity, around 17, means that ATIS requires only a 17 items vocabulary to generate all sentences ! Truly enough, some fixed constructions are used, with little variations. Theoretically, one has just to guess the type of construction and the type of variation.

5.3.b) Compared sets

Let us consider Perplexity of the different sets ;

- training set ; this one is clear, it estimates the perplexity of the model
- converging set ; this is less clear, as the factors are adapted on the set to minimize its perplexity. Difference between training and adapting perplexities can tell the reliability of adapted factors ; their generalization ability[5] .
- training set ; this perplexity has no immediate interpretation, as all statistics were taken from the set. Yet, Difference with training perplexity tells the model's generalization ability. Applying is different from training, whatever the cases, so that too well adapted models are no good besides training.

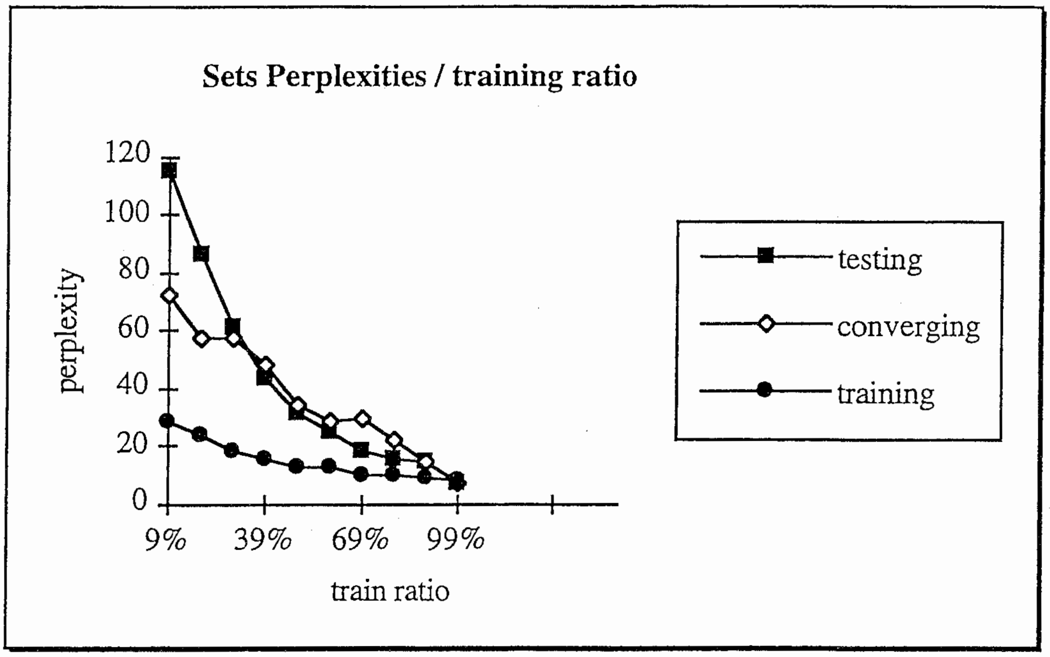


figure 5.3.3

Surprisingly enough converging perplexity is sometimes higher than testing perplexity. The converging set must have a real higher perplexity. As the set is very small (100 phrases) it can be accounted for by mere misfortune. More precise figures would be achieved by shifting systematically the sets.

5.4 Comparison

5.4.a) LDDs improve Bigrams

This Comparison is the gist of our experimentation ; are LDDs worth being added to Bigrams? We compare Bigrams performances with (LDD & Bigrams)'s ones.

In accordance with former observations, the reasonable training window is set between 80% and 90%. In those conditions the improving is pretty stable and amounts to 8%.

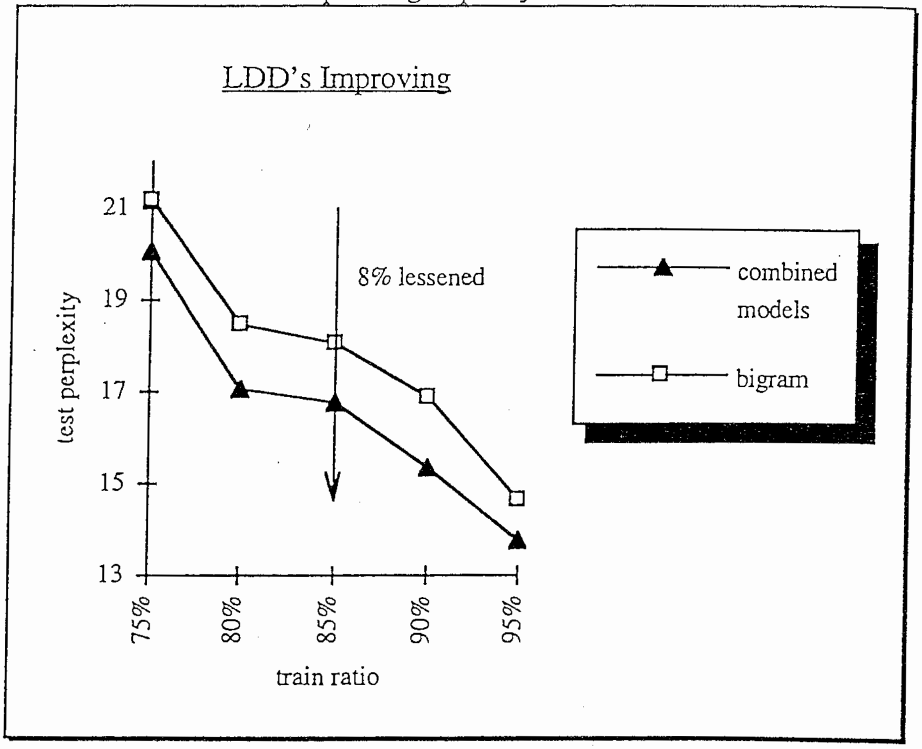


figure 5.4.1

5.4.b) Parents better than Brother ?

Parents and Brothers don't have the same part in the improving. We isolated two models; (Bigrams & Parents) and (Bigrams & Brothers), it appears, on the different sets, that Parents have more influence than Brothers.

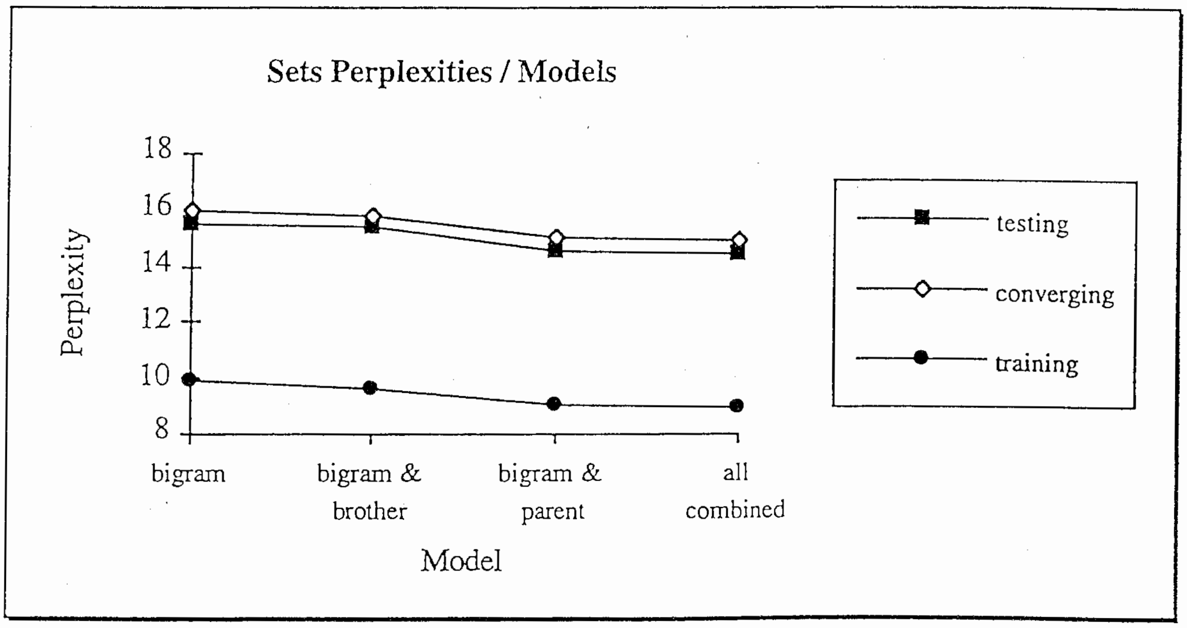


figure 5.4.2

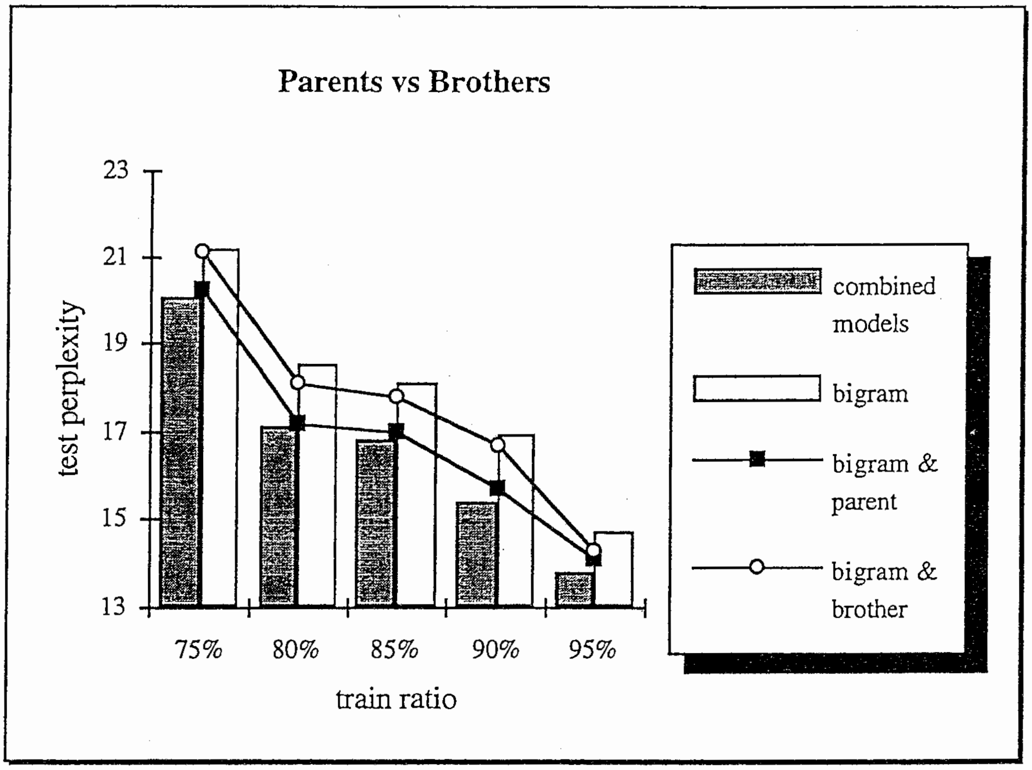


figure 5.4.3

Those figures lead to the conclusion that parents are at the improving factors, hence Parents contain Information.

5.5 Confrontation with weights

We drew conclusions from the perplexities What about the weight values? To some extent they indicates consistency.

5.5.a) about LDDs improvement

The weights found for Brothers and Parents seemed to show them as equally efficient as Bigrams in "explaining" outputs identity. From that point of view, the 8% improvement are poor.

Now, it should be kept in mind that we measured only the extra-performance on Bigrams, the poor improvement is due to redundancy or collisions in predictions ; LDD well predicted words tend to be also Bigram well predicted words.

5.5.b) about Brother / Parent comparison

Brothers weight was slightly higher than Parents', which means brothers relations were found more consistent than parents'. Thus is Brothers perplexity quite disappointing. But this poor achievement can be accounted for by scarcity of brothers, compared to parents (and worse, to bigrams) each relation is consistent enough, but there are too few to have a real impact on global recognition.

This suggests however, that corpora with longer sentences would suit brothers better, if vocabulary increase can be mastered, as long sentences are more likely to develop repetitive and symmetric structures.

5.6 Branching out

5.6.a) around those LDDs

We performed a raw experiment on brothers and parents, but some points can be developed about those very dependencies :

- incorporating transitivity and associativity potentials
- applying a context dependent combining
- using layers of word classes (grammatical, semantic etc.) in the predicting
- determining LDD without pre-parsed data

  The latter point is required for an actual application of LDDs in a speech recognition experiment. Two approaches may be figured out ;

  - real time parsing; that is quite complex, even a posteriori parsing is not automatic, yet on limited vocabulary experiments some simple structures as (NP, VP, PP) are recognize along with the uttering.

  - direct vectors' estimation; is it possible to predict next brother or parent ? Some structural recognition may be involved. The basic idea is that we are not using all information contained in the parsed structure, so there may be some avoiding of building it entirely.

5.6.b) adding other LDDs

Other kind of Long Distance Dependencies can be defined.

- without linguistic structure; some alternatives were exposed in *§3.1* .
- with linguistic structure; more information can be used from parsed sentences to define new rules, attached with extra parameters if need be, or operating on classes, for example level conscious relations or tag attached relations.

6. Conclusion

6.1 conclusion of the study

This study was a first approach to and evaluation of two particular language models, the Brother dependency and the Parent dependency. It lead to the conclusion that **Parent Model certainly captures Information, while Brother model is not adapted to the studied context.**

Now, all along the study several notions and methods were exposed. They are more than mere tools to get to the result, they reflect general points of views and directions. We present here the features we found dominant.

6.2 lesson of entropy

The Entropy approach is no doubt very promising. Basically, it stems from the idea that recognition should by ruled by information contents. On the one hand it is related to an optimal coding, on the other hand it is a very natural process of human recognition ; adapting to the expected information, put in another way **"when to expect the unexpected?"**. recognition and, generally speaking, communications, heavily depend on Information stakes. Furthermore, it is a simple an efficient way to characterize outputs, models and performance.

6.3 toward more collaboration of linguistics and statistics ?

Statistics are a powerful mean for speech processing, but there is no avoiding problems of reliability, sparseness, computation cost etc. Besides it cannot reach some bounds with no exterior hints and knowledge to guide it.

Using statistics jointly with linguistics might be an answer to statistical heaviness, in any case it seems to be the only way to break present limits of language modeling.

Now, the scope is still wide for language features to be tracked combining linguistic knowledge and statistics.

BIBLIOGRAPHY

[1] **F. Jelinek,** "Self-Organized Language Modeling for Speech Recognition", in Readings in Speech Recognition, A. Waibel, K.F. Lee, eds. 1989.

[2] **R. Rosenfeld,** " Statistical Modeling: a Maximum Entropy Approach", in *Carnegie Mellon University-Computer science-94-138* , April 1994.

[3] **P.F. Brown**, "Speech Recognition By Statistical Methods", Carnegie Mellon Univ., November 1985.

[4] **R. Isotani and S. Matsunaga,** "Speech recognition Using Stochastic Language Model Integrating Local and Global constraints", in *Proceedings of the ARPA Workshop on Human Language Technology* , March 1994.

[5] **F. Bimbot, R. Pieraccini, E. Levin, B. Atal,** "Modeles de Sequences a Horizon Variable: Multigrams", in *XXemes Journées d'Etude sur la Parole,, Trégastel* , 1st to 3rd June 1994

[6] **S.M. Katz**, "Estimation of probabilities from sparse data for the language model component of a speech recognizer", in *IEEE transactions on Acoustic , speech and signal processing,* volume ASSP-35, March 1987.

[7] **L.R. Bahl, F. Jelinek, R.L.Mercer,** "a Maximum Likelihood Approach to Continuous Speech Recognition", in *IEEE transactions on Pattern analysis and Machine Intelligence* , , vol. PAMI-5, NO.2, March 1983.

[8] **L. Rabiner, B.H Juang**, "Fundamentals of Speech Recognition", Prentice Hall eds, 1990.

[9] **L. Rabiner**, "A Tutorial on Hidden Markov Models and Selected Applications in speech Recognition", in *Proceedings of the IEEE* , vol. 77, NO. 2, February 1989

[10] **G.D. Forney Jr.**, "The Viterbi Algorithm", in *proceedings of the IEEE* , vol. 61, NO. 3, March 1973

[11] **J.P. Ueberla**, "an Extended Clustering Algorithm for statistical Language Models", *Forum Technology-DRA Malvern*, December 1994

[12] **J. Vergnes**, "A non-recursive sentence segmentation applied to parsing of linear complexity in time", *Int. Conference on New Methods in Language Processing,* Manchester UK, Sept. 1994

[13] **A. Gorin**, "On Automated Language Acquisition", AT&T Bell Lab New Jersey, October 1994, to appear in JASA.

[14] **D.Jurafsky**, "A Cognitive Model of Sentence Interpretation: the Construction Grammar approach", International Computer Science Inst&Dep of Linguistics-Univ of Calif., Berkeley, TR-93-077, December 1993

[15] **M.P. Marcus, B. Santorini, M.A Marcinkiewicz**, "Building a large annotated corpus of English: the Penn Treebank", Pennsylvania Univ-Dep of computer and Information sciences, Northwestern Univ-Dep of Linguistics, May 1991.

[16] **B. Santorini,** "Bracketing Guidelines for Penn Treebank Project", idem

[17] **Y. Lepages L. Fais**, "Syntactic trees for sentences of ten dialogues", ATR-ITL-Technical Report, January 1994.

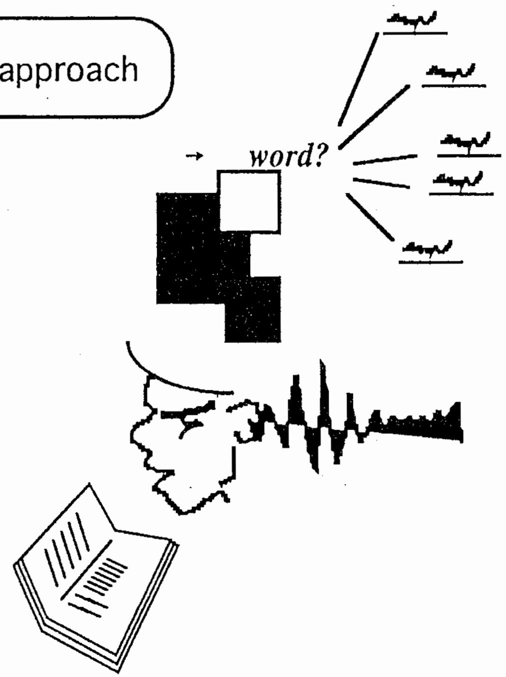
Annex I

Slides used for presentation

AIM OF THE STUDY

Improving speech recognition with language modeling

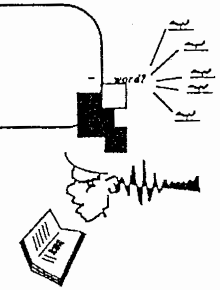
- speech recognition using language models?
  - why? → statistical approach
  - how? → word history : classical models
- another history...
  - hypothese → long distance dependencies
  - implement → combining
- Improving?
  - evaluate → Information Theory
  - discuss → results



Ariane Halber, ATR Interpreting Telecommunications Research Lab.

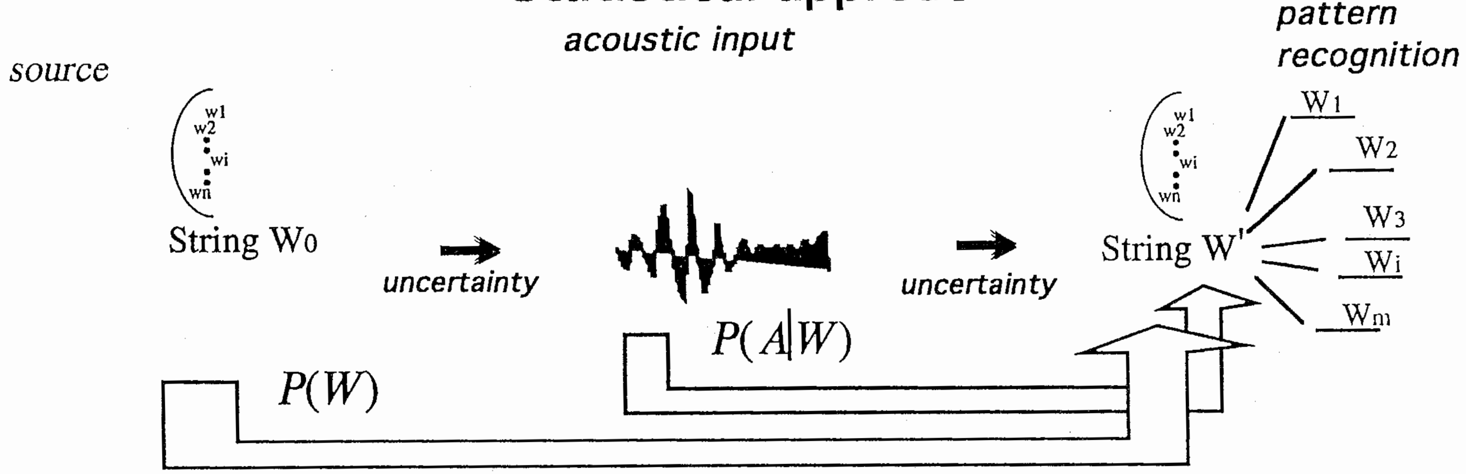
ATR

language modeling for speech recognition



why?

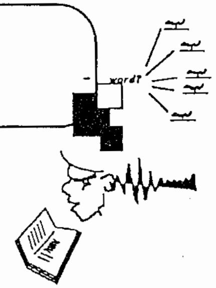
Statistical approach



$$\Rightarrow \quad \boxed{W' = \arg\max_{W} P(W)\,P(A|W)}$$

Ariane Halber, ATR Interpreting Telecommunications Research Lab.

language modeling for speech recognition



why?
Statistical approach

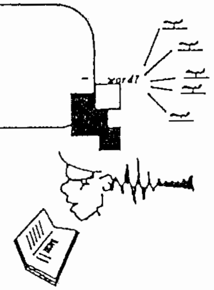
most likely String $S' = (w_1, w_2, .., w_n)$ given acoustic evidence A : $P(W'|A) = \max_W P(W|A)$

Bayes' Fromula : $P(W|A) = \dfrac{P(W)\,P(A|W)}{P(A)}$

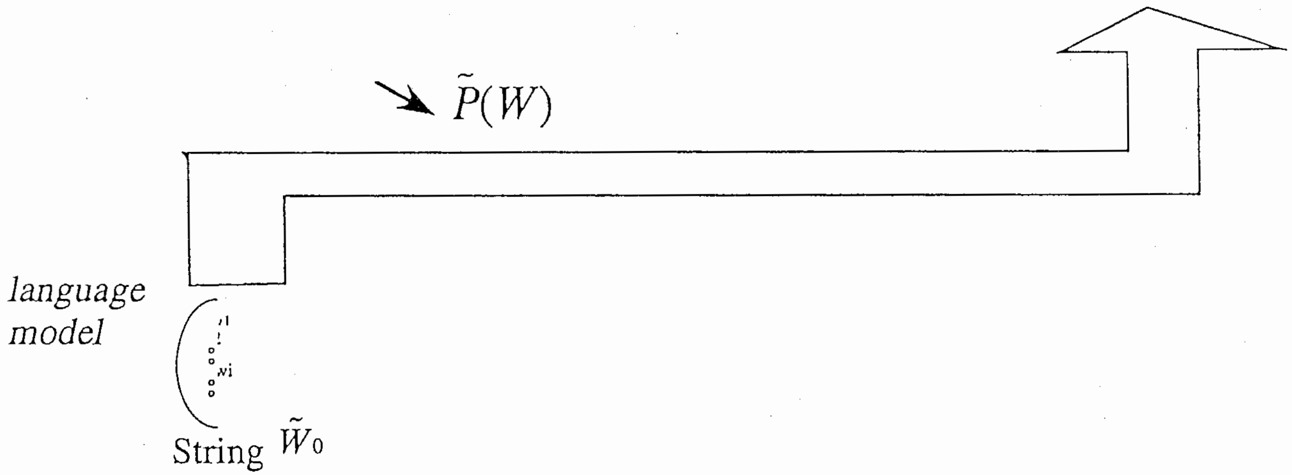
$$\Rightarrow \quad P(W'|A) = \max_W \frac{P(W)\,P(A|W)}{P(A)}$$

ATR

language modeling for speech recognition

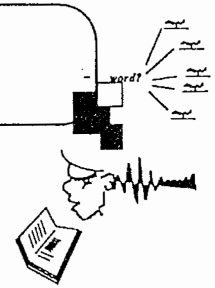


why?
Statistical approach



Ariane Halber, ATR Interpreting Telecommunications Research Lab.

language modeling for speech recognition



how?
word's history ~ classical models

General form:

$$P(W) = \prod_{i=1}^{n} P\left(w_i \middle| \overbrace{w_1,..,w_{i-1}}^{history}\right)$$

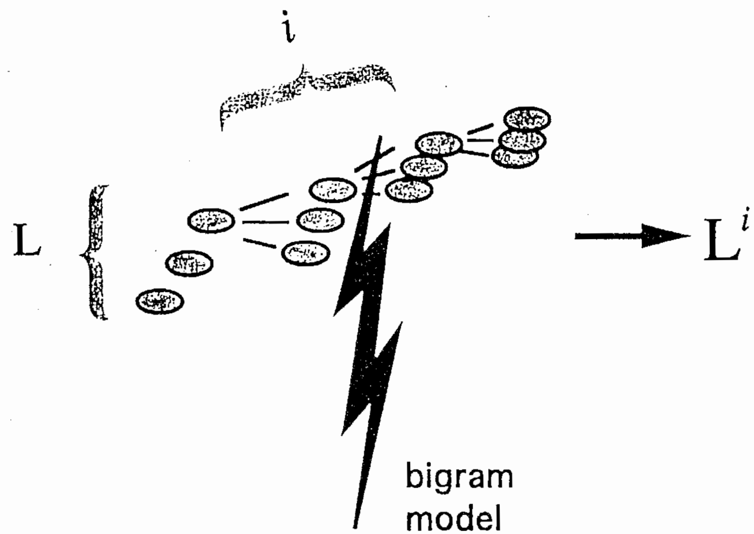
Problem : exponential

Language model : **histories classes**

$$P(W) = \prod_{i=1}^{n} P\left(w_i \middle| S\left[w_1,..,w_{i-1}\right]\right)$$

class caracterisation = kept "information"

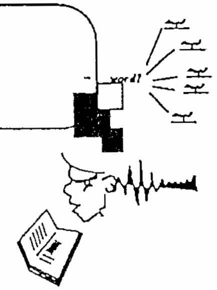
N_Grams : histories are caracterized by their last N words



Ariane Halber, ATR Interpreting Telecommunications Research Lab.

ATR

language modeling for speech recognition



how?
N_Grams and their limits

Issue: computing $P(W) = \prod_{i=1}^{n} P(w_i | w_{i-1}, .., w_{i-N})$

training corpus ⟶ $(w_1, w_2,,,, w_N)$ exponential

- huge number of parameter ⟶ N=1 or 2
- problem of sparseness ⟶ N=1 or 2
- constaint of contiguity

efficient model but : heavy, doesn't take context in account, nor longer memory

Ariane Halber, ATR Interpreting Telecommunications Research Lab.

ATR

long distance dependencies

alternatives

hypothesis: information from distant word has predictiv power

reaching far connections between words:

- fixed distance (restrictiv / hypothesis possibilities)
- computed distance (for one couple then)
- dist. independant, automatic organizing (computationnaly heavy)
- dist. independant, adapted to context

using context information swiftly, introducing another knowledge

Ariane Halber, ATR Interpreting Telecommunications Research Lab.

longues distance dependencies

linguistic knowledge: the parsed corpus

bidimensional:

tree structure: Tags as Nodes

Words as leaves

syntactic

a set of 15 Tags , not so much thanks to the structural information

clustered

```
( (S (NP *)
  (VP Show
    (NP me)
    (NP (NP all)
      the nonstop flights
      (PP (PP from
            (NP Dallas))
          (PP to
            (NP Denver)))
      (ADJP early
         (PP in
           (NP the morning)))))
  .)
)
```

Ariane Halber, ATR Interpreting Telecommunications Research Lab.

longues distance dependencies

dependency rules

directly steming from the tree structure:

TAG brothers

recursive relation

Proposition Father

a word who triggers a Node

```
( (S (NP *)
   (VP Show
      (NP me)
      (NP (NP all)
         the nonstop flights
         (PP (PP from
               (NP Dallas))
            (PP to
               (NP Denver)))
         (ADJP early
            (PP in
               (NP the morning)))))
   .)
)
```

ATR

long distance dependencies

Brothers

---

97 [can]
--> [does]

---

99 [capacity]
--> [weight]

---

108 [cheapest]
--> [available]

---

110 [city]
--> [PLACE]

---

221 [from]
--> [back]
--> [after]
--> [into]
--> [to]

---

283 [leaving]
--> [arriving]
--> [returning]
--> [going]

Parents

29 [after]
--> [days]
--> [p.m]
--> [eight]
--> [o'clock]
--> [five]
--> [noon]
--> [a.m]
--> [p.m.]
--> [NUMmix]

---

32 [again]
--> [NUMth]
--> [MONTH]

---

65 [arriving]
--> [noon]
--> [o'clock]
--> [twelve]
--> [approximately]
--> [before]
--> [into]
--> [at]

Now how to assess the worth of a dependency ?

what about its potentality?

its optimal use?

the recogniton task

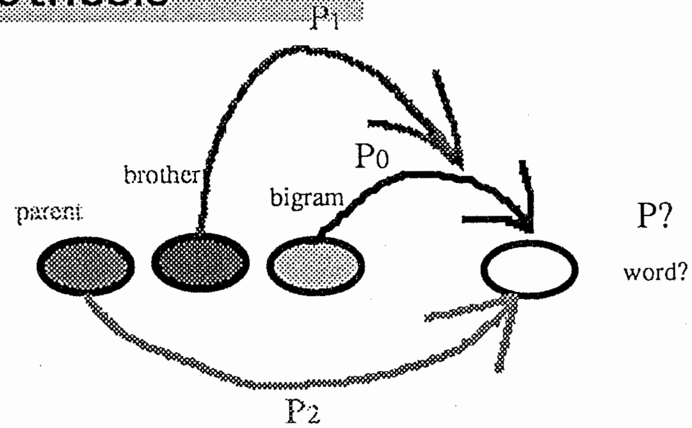
capturing Information...

ATR

recognition task

context & hypothesis

we want to evaluate ldd's potentialities



→ Structure = ideal case

knowing bigram, brother, parent, ie
having a sentence squeletton

→ Data = "Atis", specialized but sparse

making up for sparseness by smoothing statistics

→ Prediction = combining the 3 informations consistently:

linear interpolation ; EM algorithm

Ariane Halber, ATR Interpreting Telecommunications Research Lab.

recognition task

problem : sparseness

atis file

1382 sentences (16 273 words)	found bigrams : 2172 new bigrams : 537
1151 sentences for training	found parents : 647 new parents : 277
231 sentences for testing	found brothers : 99 new brothers : 21

- 3/4 for training ⟶ still 1 bigram out of 5 is unknown ⟶ smoothing bigrams
- same problem for brothers and parents ⟶ smoothing too? same method?
- could ldd rather mislead recognition because of their rarity? ⟶ problem of combining

recognition task

smoothing

training data

maximum likelihood estimate: $P_{ML}=0$ for unseen sequences, ie existing strings are said impossible.

$\Rightarrow$ other statistic : Turing estimate , $P_T(W) = d_r P_{ML}(W)$, $d_r = \frac{r+1}{r}\frac{n_{r+1}}{n_r}$ discounting ratio

$$r = C(W)\ ,\ n_r = card\left\{W \middle| C(W) = r\right\}$$

manyfold

→ discounts probabilities of existant sequences in training data

→ affects proba to unseen sequences
- uniformely
- or according to a certain criterium

eg. we did according to the probability (estimate) of the lower level

→ keep unaffected the high frequency sequences (they're less "fantomatic")

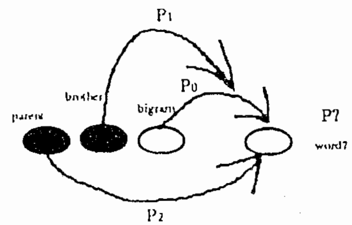
(→ if need be adjust with normalizing factor, make sure it is still a statistic)

ATR

recognition task

combining

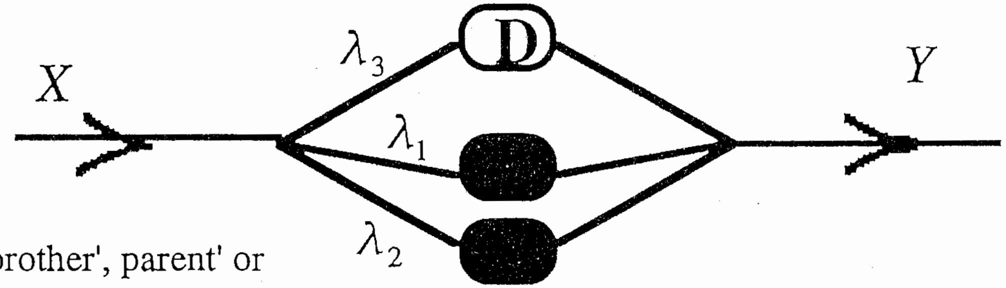
$$P_{big}(w|big),\ P_{bro}(w|bro),\ P_{par}(w|par)$$

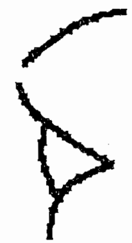


correlation between different distributions?

linear interpolation $(1-\lambda_1-\lambda_2)\,P_{big}(w|big) + \lambda_1\, P_{bro}(w|bro) + \lambda_2\, P_{par}(w|par)$

according to the model :





λ_1 λ_2 λ_3 statistics of use of brother', parent' or bigram's Distribution.

maximize $E_\theta\left(\log P_{\hat{\theta}}(x,y)|y\right)$

let $P_\theta(x,y) = e^{T(x,y)}$

$\Rightarrow$ maximize $E_\lambda\left[T(x,y)\right]$

→ converge iteratively with EM algorithm: assuming initial value maximizing Expectation

intuitively : converges toward its relativ weight; $\hat{\lambda}_1 = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \frac{\hat{\lambda}_1 P_1(y_i)}{\hat{\lambda}_1 P_1(y_i) + \hat{\lambda}_2 P_2(y_i)}$

Ariane Halber, ATR Interpreting Telecommunications Research Lab.

ATR

RESULTS

brothers

```
--> [from]       mutualcount= 1     mutualinfo= -0.00025C
--> [back]       mutualcount= 2     mutualinfo= 0.000414
--> [after]      mutualcount= 1     mutualinfo= -0.00002C
--> [into]       mutualcount= 3     mutualinfo= 0.000653
--> [on]         mutualcount= 4     mutualinfo= -0.000433
--> [at]mutualcount= 1     mutualinfo= -0.000085
--> [to]mutualcount= 371   mutualinfo= 0.090197
----------------------------
[leave]          count=20      (proba=0.001227)
--> [are]        mutualcount= 1     mutualinfo= 0.000184
--> [arrive]     mutualcount= 5     mutualinfo= 0.002273
--> [come]       mutualcount= 1     mutualinfo= 0.000593
----------------------------
[leaves]         count=10      (proba=0.000614)
--> [arrives]    mutualcount= 3     mutualinfo= 0.001780
----------------------------
[leaving]        count=70      (proba=0.004296)
--> [going]      mutualcount= 1     mutualinfo= 0.000249
--> [departing]  mutualcount= 1     mutualinfo= 0.000184
--> [arriving]   mutualcount= 20    mutualinfo= 0.008469
--> [returning]  mutualcount= 9     mutualinfo= 0.003439
----------------------------
```

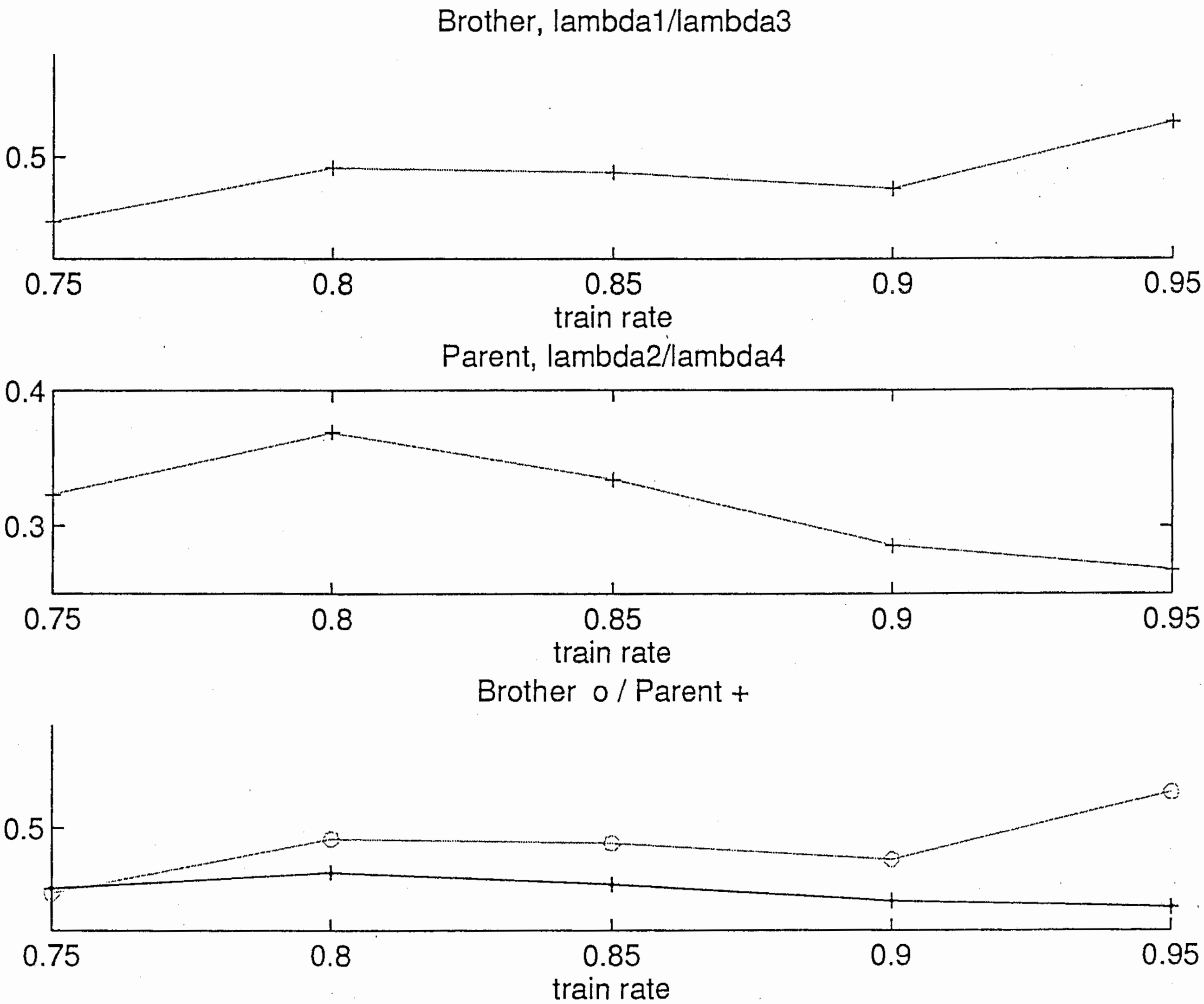
dependencies & mutual information

the consistent and
the misleading

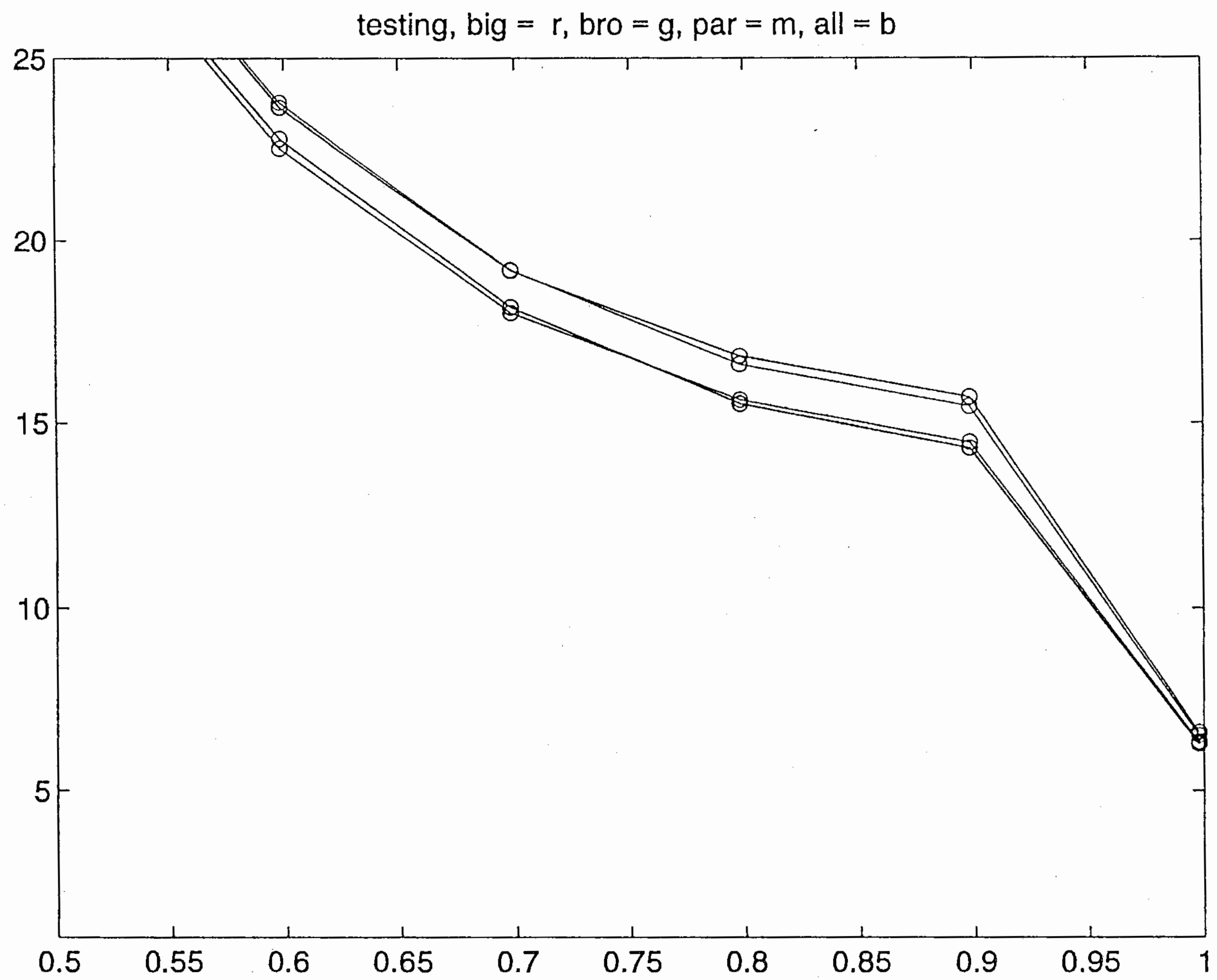
power of discrete dependencies?

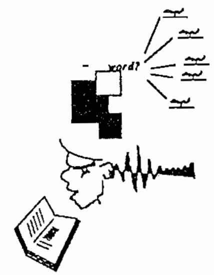
training 85%

	potential division of perplexity	division of bigrams perplexity
bigrams	13.2	1
brothers	1.2	1.02
parents	3.7	1.1


Brother, lambda1/lambda3
0.5
0.75
0.8
0.85
0.9
0.95
train rate
Parent, lambda2/lambda4
0.4
0.3
0.75
0.8
0.85
0.9
0.95
train rate
Brother o / Parent +
0.5
0.75
0.8
0.85
0.9
0.95
train rate

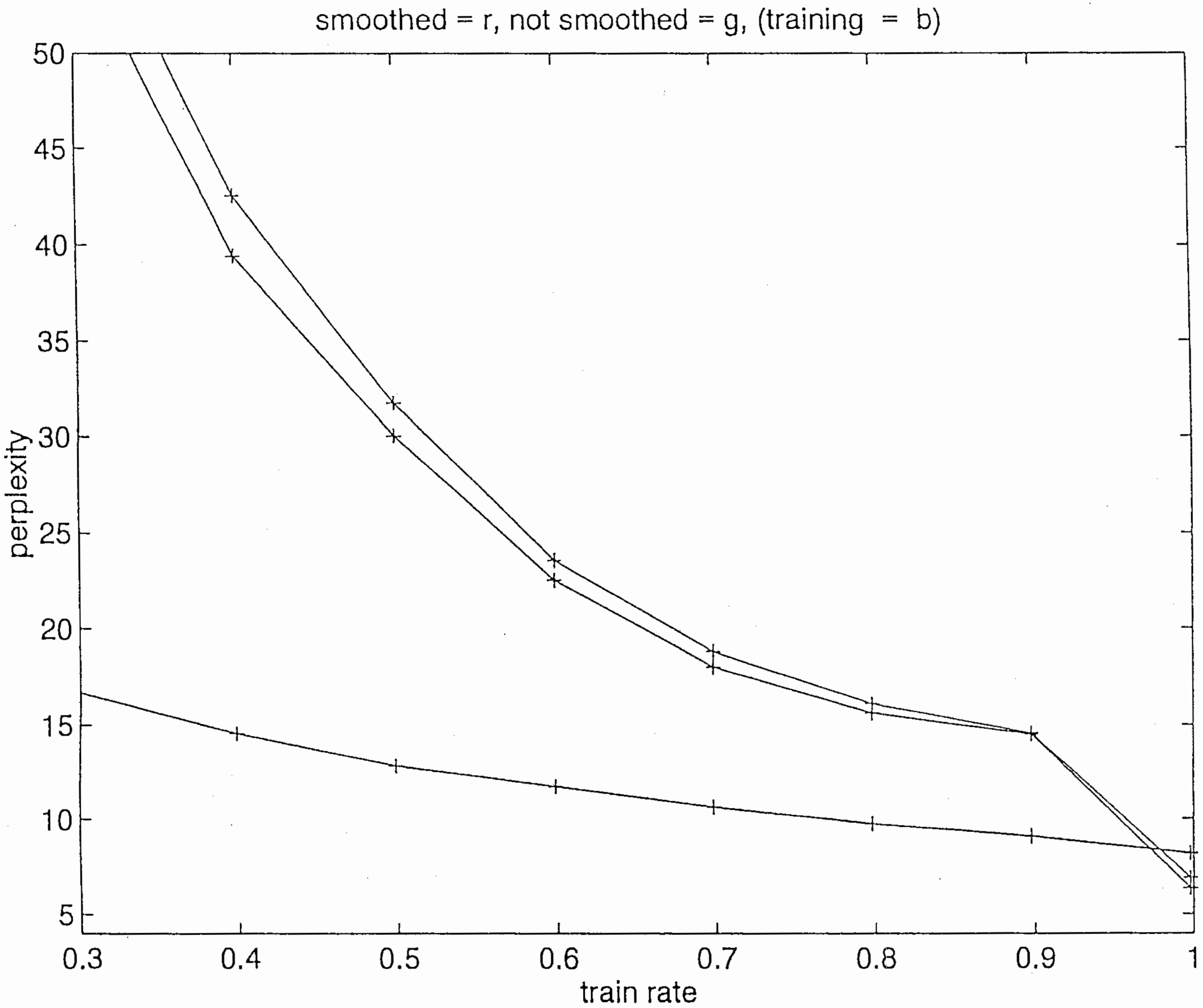
ATR



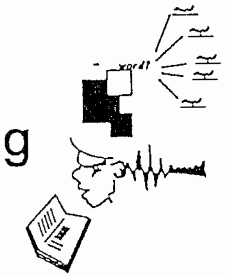


Ariane Halber, ATR Interpreting Telecommunications Research Lab.

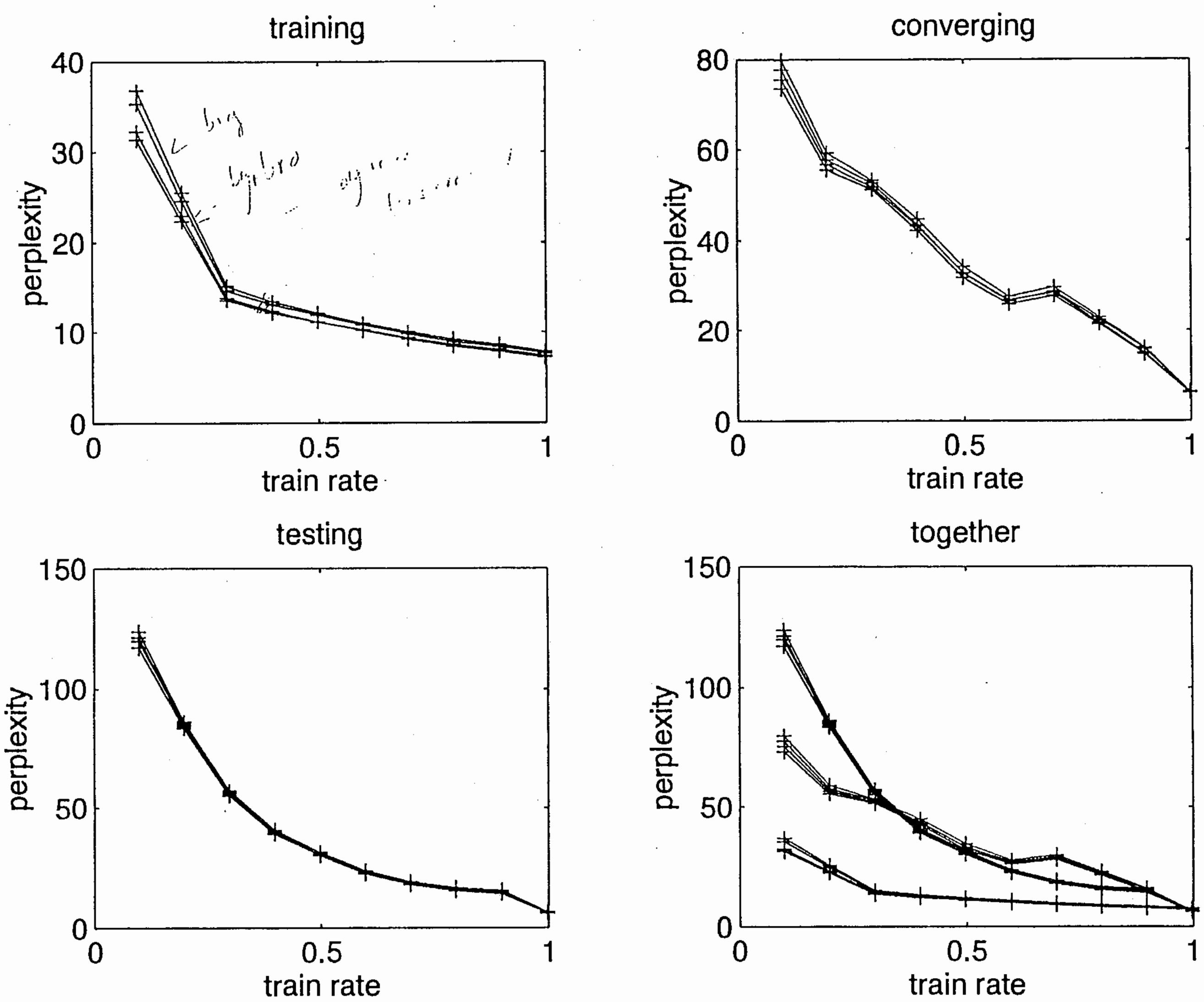
ATR

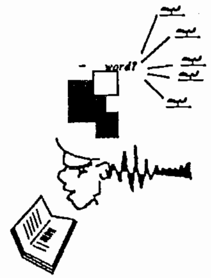

smoothed = r, not smoothed = g, (training = b)
perplexity
train rate
50
45
40
35
30
25
20
15
10
5
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

ng



ATR





Ariane Halber, ATR Interpreting Telecommunications Research Lab.

Annex II

some brother dependencies

```
            a
          * * * * * * *

---------------------------
1        [NUM    ]

count=535        (proba=0.036237)

 --> [bwi]               count= 7        mutualinfo= 0.000134
 --> [NUM    ]           count= 535      mutualinfo= -0.000444
 --> [noon]              count= 36       mutualinfo= -0.000026
 --> [fare]              count= 132      mutualinfo= -0.000153
 --> [PLACE ]            count= 1416     mutualinfo= -0.000634
 --> [MONTH ]            count= 190      mutualinfo= -0.000189
 --> [time]              count= 19       mutualinfo= 0.000036
---------------------------
4        [PLACE ]

count=1416       (proba=0.095909)

 --> [d.c]               count= 1        mutualinfo= 0.000229
 --> [fare]              count= 132      mutualinfo= -0.000361
 --> [worth]             count= 37       mutualinfo= -0.000124
 --> [NUMth ]            count= 191      mutualinfo= -0.000284
 --> [NUM    ]           count= 535      mutualinfo= -0.000385
 --> [PLACE ]            count= 1416     mutualinfo= -0.002549
---------------------------
5        [MONTH ]

count=190        (proba=0.012869)

 --> [number]            count= 90       mutualinfo= -0.000014
 --> [seats]             count= 21       mutualinfo= 0.000128
 --> [the]               count= 641      mutualinfo= 0.000439
 --> [NUMth ]            count= 191      mutualinfo= 0.000190
 --> [NUM    ]           count= 535      mutualinfo= 0.006343
---------------------------
6        [DAY    ]

count=29         (proba=0.001964)

 --> [the]               count= 641      mutualinfo= 0.000090
 --> [NUM    ]           count= 535      mutualinfo= -0.000005
 --> [MONTH ]            count= 190      mutualinfo= 0.000927
---------------------------
17       [*]

count=700        (proba=0.047413)

 --> [describe]          count= 12       mutualinfo= 0.000055
 --> [*]                 count= 700      mutualinfo= -0.000342
 --> [NUM    ]           count= 535      mutualinfo= -0.000316
---------------------------
18       [*pseudo-attach*]

count=41         (proba=0.002777)

 --> [for]               count= 215      mutualinfo= 0.000236
 --> [*pseudo-attach*]          count= 41        mutualinfo= 0.002324
---------------------------
23       [abbreviation]

count=12         (proba=0.000813)

 --> [r]                 count= 6        mutualinfo= 0.000520
 --> [l]                 count= 6        mutualinfo= 0.000520
 --> [a]                 count= 216      mutualinfo= 0.000170
---------------------------

36       [after]

count=38         (proba=0.002574)

 --> [with]              count= 29       mutualinfo= 0.000643
 --> [from]              count= 467      mutualinfo= -0.000018
 --> [on]                count= 377      mutualinfo= 0.000554
---------------------------
37       [afternoon]

count=10         (proba=0.000677)

 --> [around]            count= 5        mutualinfo= 0.000556
---------------------------
47       [airline]

count=28         (proba=0.001897)

 --> [us]                count= 13       mutualinfo= 0.000362
 --> [co]                count= 10       mutualinfo= 0.000387
---------------------------
48       [airlines]

count=86         (proba=0.005825)

 --> [day]               count= 12       mutualinfo= 0.000260
 --> [MONTH ]            count= 190      mutualinfo= -0.000010
 --> [class]             count= 153      mutualinfo= 0.000011
 --> [NUM    ]           count= 535      mutualinfo= 0.001814
---------------------------
51       [airport]

count=60         (proba=0.004064)

 --> [bwi]               count= 7        mutualinfo= 0.000348
 --> [den]               count= 1        mutualinfo= 0.000538
 --> [city]              count= 6        mutualinfo= 0.000861
---------------------------
54       [all]

count=115        (proba=0.007789)

 --> [description]              count= 5         mutualinfo= 0.000317
---------------------------
58       [american]

count=41         (proba=0.002777)

 --> [number]            count= 90       mutualinfo= 0.001464
 --> [NUM    ]           count= 535      mutualinfo= -0.000039
---------------------------
70       [are]

count=97         (proba=0.006570)

 --> [are]               count= 97       mutualinfo= 0.000044
---------------------------
75       [arrangements]

count=10         (proba=0.000677)

 --> [*pseudo-attach*]          count= 41        mutualinfo= 0.000350
 --> [airlines]          count= 86       mutualinfo= 0.000278
---------------------------
79       [arriving]

count=39         (proba=0.002642)

 --> [flying]            count= 10       mutualinfo= 0.000355
---------------------------
```

```
81        [at]
count=77         (proba=0.005215)
 --> [from]                 count= 467        mutualinfo= -0.000087
 --> [before]               count= 32         mutualinfo= 0.000175
 --> [at]                   count= 77         mutualinfo= 0.000089
------------------------------
86        [b]
count=15         (proba=0.001016)
 --> [y]                    count= 17         mutualinfo= 0.000397
------------------------------
88        [back]
count=6 (proba=0.000406)
 --> [on]                   count= 377        mutualinfo= 0.000183
------------------------------
92        [before]
count=32         (proba=0.002167)
 --> [on]                   count= 377        mutualinfo= 0.000020
 --> [from]                 count= 467        mutualinfo= -0.000001
------------------------------
102       [book]
count=138        (proba=0.009347)
 --> [leaving]              count= 65         mutualinfo= 0.000049
------------------------------
114       [by]
count=16         (proba=0.001084)
 --> [from]                 count= 467        mutualinfo= 0.000269
------------------------------
133       [city]
count=6 (proba=0.000406)
 --> [PLACE ]               count= 1416       mutualinfo= 0.000243
------------------------------
135       [class]
count=153        (proba=0.010363)
 --> [y]                    count= 17         mutualinfo= 0.000170
 --> [q]                    count= 8          mutualinfo= 0.000243
 --> [qw]                   count= 10         mutualinfo= 0.000222
 --> [MONTH ]               count= 190        mutualinfo= -0.000066
 --> [also]                 count= 16         mutualinfo= 0.000176
 --> [service]              count= 19         mutualinfo= 0.000159
------------------------------
136       [classes]
count=24         (proba=0.001626)
 --> [and]                  count= 177        mutualinfo= 0.000122
------------------------------
138       [co]
count=10         (proba=0.000677)
 --> [number]               count= 90         mutualinfo= 0.000273
------------------------------
140       [code]
count=43         (proba=0.002912)
 --> [NUM    ]              count= 535        mutualinfo= -0.000043
 --> [qx]                   count= 13         mutualinfo= 0.000320
 --> [1]                    count= 6          mutualinfo= 0.000395
 --> [a]                    count= 216        mutualinfo= 0.000045
------------------------------
157       .[continental]
count=5 (proba=0.000339)
 --> [NUM    ]              count= 535        mutualinfo= 0.000167
------------------------------
171       [date]
count=10         (proba=0.000677)
 --> [MONTH ]               count= 190        mutualinfo= 0.000200
------------------------------
172       [day]
count=12         (proba=0.000813)
 --> [NUM    ]              count= 535        mutualinfo= 0.000566
------------------------------
173       [days]
count=8 (proba=0.000542)
 --> [week]                 count= 2          mutualinfo= 0.000667
------------------------------
175       [dc9]
count=1 (proba=0.000068)
 --> [NUM    ]              count= 535        mutualinfo= 0.000324
------------------------------
178       [delta]
count=86         (proba=0.005825)
 --> [number]               count= 90         mutualinfo= 0.000262
 --> [us]                   count= 13         mutualinfo= 0.000252
 --> [NUM    ]              count= 535        mutualinfo= 0.000231
 --> [delta]                count= 86         mutualinfo= 0.000271
------------------------------
181       [departing]
count=29         (proba=0.001964)
 --> [arriving]             count= 39         mutualinfo= 0.002042
------------------------------
183       [departs]
count=4 (proba=0.000271)
 --> [arrives]              count= 6          mutualinfo= 0.000628
------------------------------
235       [f28]
count=1 (proba=0.000068)
 --> [NUMmix]               count= 45         mutualinfo= 0.000566
------------------------------
238       [fare]
count=132        (proba=0.008941)
 --> [*pseudo-attach*]              count= 41         mutualinfo= 0.000098
------------------------------
```

```
244      [field]

count=3 (proba=0.000203)

 --> [day]              count= 12      mutualinfo= 0.000588
----------------------------
249      [flight]

count=419       (proba=0.028380)

 --> [number]           count= 90      mutualinfo= 0.000501
 --> [PLACE ]           count= 1416    mutualinfo= -0.000361
 --> [fare]             count= 132     mutualinfo= -0.000129
 --> [NUM   ]           count= 535     mutualinfo= -0.000544
 --> [and]              count= 177     mutualinfo= -0.000158
 --> [afternoon]                count= 10      mutualinfo= 0.000123
 --> [class]            count= 153     mutualinfo= -0.000152
 --> [person]           count= 22      mutualinfo= 0.000046
----------------------------
250      [flights]

count=304       (proba=0.020591)

 --> [flights]          count= 304     mutualinfo= -0.000179
 --> [fares]            count= 58      mutualinfo= -0.000017
 --> [fare]             count= 132     mutualinfo= -0.000098
----------------------------
256      [for]

count=215       (proba=0.014562)

 --> [at]               count= 77      mutualinfo= 0.000113
 --> [for]              count= 215     mutualinfo= -0.000088
 --> [*pseudo-attach*]          count= 41      mutualinfo= 0.000050
 --> [to]               count= 686     mutualinfo= -0.000225
 --> [from]             count= 467     mutualinfo= -0.000240
 --> [between]          count= 22      mutualinfo= 0.000111
 --> [on]               count= 377     mutualinfo= 0.000586
----------------------------
261      [from]

count=467       (proba=0.031631)

 --> [back]             count= 6       mutualinfo= 0.000460
 --> [after]            count= 38      mutualinfo= -0.000018
 --> [into]             count= 8       mutualinfo= 0.000404
 --> [on]               count= 377     mutualinfo= -0.000427
 --> [in]               count= 123     mutualinfo= -0.000076
 --> [*pseudo-attach*]          count= 41      mutualinfo= -0.000025
 --> [at]               count= 77      mutualinfo= -0.000087
 --> [to]               count= 686     mutualinfo= 0.046579
----------------------------
269      [give]

count=73        (proba=0.004944)

 --> [show]             count= 254     mutualinfo= -0.000022
----------------------------
270      [go]

count=8 (proba=0.000542)

 --> [make]             count= 41      mutualinfo= 0.000372
 --> [put]              count= 1       mutualinfo= 0.000735
----------------------------
271      [going]

count=14        (proba=0.000948)

 --> [and]              count= 177     mutualinfo= 0.000174
```

```
----------------------------
296      [i]

count=243       (proba=0.016459)

 --> [*]                count= 700     mutualinfo= -0.000239
 --> [i]                count= 243     mutualinfo= 0.000000
----------------------------
298      [in]

count=123       (proba=0.008331)

 --> [to]               count= 686     mutualinfo= -0.000170
 --> [at]               count= 77      mutualinfo= 0.000043
 --> [by]               count= 16      mutualinfo= 0.001330
 --> [after]            count= 38      mutualinfo= 0.000112
 --> [*pseudo-attach*]          count= 41      mutualinfo= 0.000105
 --> [on]               count= 377     mutualinfo= -0.000112
----------------------------
301      [information]

count=60        (proba=0.004064)

 --> [PLACE ]           count= 1416    mutualinfo= -0.000191
----------------------------
304      [inquiry]

count=6 (proba=0.000406)

 --> [flight]           count= 419     mutualinfo= 0.000173
----------------------------
309      [into]

count=8 (proba=0.000542)

 --> [on]               count= 377     mutualinfo= 0.000155
 --> [at]               count= 77      mutualinfo= 0.000310
 --> [by]               count= 16      mutualinfo= 0.001714
----------------------------
311      [it]

count=26        (proba=0.001761)

 --> [i]                count= 243     mutualinfo= 0.000083
 --> [it]               count= 26      mutualinfo= 0.000301
----------------------------
332      [leave]

count=20        (proba=0.001355)

 --> [come]             count= 1       mutualinfo= 0.000645
 --> [arrive]           count= 24      mutualinfo= 0.001326
----------------------------
333      [leaves]

count=8 (proba=0.000542)

 --> [gets]             count= 2       mutualinfo= 0.000667
 --> [arrives]          count= 6       mutualinfo= 0.002001
----------------------------
334      [leaving]

count=65        (proba=0.004403)

 --> [going]            count= 14      mutualinfo= 0.000680
 --> [departing]                count= 29      mutualinfo= 0.000201
 --> [arriving]         count= 39      mutualinfo= 0.009298
 --> [returning]                count= 24      mutualinfo= 0.002368
----------------------------
344      [list]
```

```
count=53        (proba=0.003590)

 --> [show]               count= 254      mutualinfo= 0.000009
 --> [list]               count= 53       mutualinfo= 0.000809
-----------------------------
361      [max]

count=2 (proba=0.000135)

 --> [min]                count= 4        mutualinfo= 0.000735
-----------------------------
366      [me]

count=287       (proba=0.019439)

 --> [codes]              count= 22       mutualinfo= 0.000083
 --> [NUM    ]            count= 535      mutualinfo= -0.000364
 --> [the]                count= 641      mutualinfo= -0.000446
 --> [capacity]           count= 7        mutualinfo= 0.001321
 --> [description]                count= 5        mutualinfo= 0.000228
 --> [performance]                count= 1        mutualinfo= 0.000385
 --> [price]              count= 15       mutualinfo= 0.000683
 --> [help]               count= 4        mutualinfo= 0.001540
 --> [only]               count= 12       mutualinfo= 0.000142
 --> [departing]                  count= 29       mutualinfo= 0.000056
 --> [charges]            count= 2        mutualinfo= 0.000317
 --> [PLACE ]             count= 1416     mutualinfo= -0.000512
 --> [transportation]             count= 73       mutualinfo= 0.000405
 --> [cost]               count= 32       mutualinfo= 0.000727
 --> [reservations]               count= 29       mutualinfo= 0.000056
 --> [availability]               count= 2        mutualinfo= 0.000317
 --> [type]               count= 21       mutualinfo= 0.000088
 --> [list]               count= 53       mutualinfo= 0.001033
 --> [airlines]           count= 86       mutualinfo= -0.000050
 --> [space]              count= 1        mutualinfo= 0.000385
 --> [all]                count= 115      mutualinfo= 0.001970
 --> [flights]            count= 304      mutualinfo= 0.003287
 --> [listing]            count= 15       mutualinfo= 0.000120
 --> [fares]              count= 58       mutualinfo= 0.000495
 --> [number]             count= 90       mutualinfo= -0.000055
 --> [costs]              count= 6        mutualinfo= 0.000210
 --> [numbers]            count= 4        mutualinfo= 0.000250
 --> [and]                count= 177      mutualinfo= -0.000121
 --> [such]               count= 1        mutualinfo= 0.000385
 --> [fare]               count= 132      mutualinfo= -0.000049
 --> [service]            count= 19       mutualinfo= 0.000097
 --> [information]                count= 60       mutualinfo= 0.004451
-----------------------------
377      [midnight]

count=1 (proba=0.000068)

 --> [four]               count= 5        mutualinfo= 0.000781
-----------------------------
401      [nonstop]

count=39        (proba=0.002642)

 --> [first]              count= 53       mutualinfo= 0.000520
-----------------------------
406      [number]

count=90        (proba=0.006096)

 --> [airlines]           count= 86       mutualinfo= 0.000262
 --> [class]              count= 153      mutualinfo= 0.000149
 --> [number]             count= 90       mutualinfo= 0.001402
-----------------------------
409      [of]

count=191       (proba=0.012937)

 --> [to]                 count= 686      mutualinfo= -0.000291
 --> [on]                 count= 377      mutualinfo= -0.000155
 --> [for]                count= 215      mutualinfo= -0.000100
-----------------------------
415      [okay]

count=9 (proba=0.000610)

 --> [fine]               count= 1        mutualinfo= 0.000723
-----------------------------
416      [on]

count=377       (proba=0.025535)

 --> [under]              count= 24       mutualinfo= 0.000048
 --> [*pseudo-attach*]            count= 41       mutualinfo= 0.000127
 --> [from]               count= 467      mutualinfo= -0.000425
 --> [with]               count= 29       mutualinfo= 0.000029
 --> [for]                count= 215      mutualinfo= 0.000435
 --> [on]                 count= 377      mutualinfo= 0.000143
-----------------------------
417      [one]

count=43        (proba=0.002912)

 --> [NUM    ]            count= 535      mutualinfo= -0.000043
 --> [*]                  count= 700      mutualinfo= -0.000070
-----------------------------
429      [out]

count=8 (proba=0.000542)

 --> [to]                 count= 686      mutualinfo= 0.000329
-----------------------------
443      [person]

count=22        (proba=0.001490)

 --> [class]              count= 153      mutualinfo= 0.000144
-----------------------------
454      [possibly]

count=1 (proba=0.000068)

 --> [on]                 count= 377      mutualinfo= 0.000358
-----------------------------
468      [qw]

count=10        (proba=0.000677)

 --> [qx]                 count= 13       mutualinfo= 0.000462
-----------------------------
469      [qx]

count=13        (proba=0.000881)

 --> [y]                  count= 17       mutualinfo= 0.000411
-----------------------------
485      [reservation]

count=31        (proba=0.002100)

 --> [class]              count= 153      mutualinfo= 0.000357
 --> [*pseudo-attach*]            count= 41       mutualinfo= 0.000240
 --> [but]                count= 2        mutualinfo= 0.000535
-----------------------------
487      [reserve]
```

```
count=23        (proba=0.001558)

 --> [reserve]              count= 23        mutualinfo= 0.000325
----------------------------
491     [restriction]

count=28        (proba=0.001897)

 --> [vu]                   count= 13        mutualinfo= 0.000362
----------------------------
508     [seats]

count=21        (proba=0.001422)

 --> [fare]                 count= 132       mutualinfo= 0.000163
 --> [MONTH ]               count= 190       mutualinfo= 0.000391
 --> [NUM   ]               count= 535       mutualinfo= 0.000027
 --> [flight]               count= 419       mutualinfo= 0.001685
----------------------------
509     [second]

count=2 (proba=0.000135)

 --> [*pseudo-attach*]              count= 41        mutualinfo= 0.000507
----------------------------
516     [service]

count=19        (proba=0.001287)

 --> [PLACE ]               count= 1416      mutualinfo= -0.000059
----------------------------
558     [symbol]

count=2 (proba=0.000135)

 --> [vu]                   count= 13        mutualinfo= 0.000620
 --> [and]                  count= 177       mutualinfo= 0.000365
----------------------------
560     [t]

count=357       (proba=0.024180)

 --> [or]                   count= 16        mutualinfo= 0.000093
----------------------------
573     [that]

count=79        (proba=0.005351)

 --> [fare]                 count= 132       mutualinfo= 0.000034
 --> [*]                    count= 700       mutualinfo= -0.000129
----------------------------
574     [the]

count=641       (proba=0.043416)

 --> [DAY   ]               count= 29        mutualinfo= -0.000023
----------------------------
578     [there]

count=43        (proba=0.002912)

 --> [layovers]             count= 3         mutualinfo= 0.001062
 --> [flights]              count= 304       mutualinfo= 0.000012
 --> [reservation]                  count= 31        mutualinfo= 0.000235
 --> [airlines]             count= 86        mutualinfo= 0.000135
 --> [any]                  count= 28        mutualinfo= 0.000245
 --> [ground]               count= 69        mutualinfo= 0.000449
 --> [transportation]               count= 73        mutualinfo= 0.004729
 --> [a]                    count= 216       mutualinfo= 0.000045

 --> [restrictions]                 count= 20        mutualinfo= 0.000278
 --> [conditions]                   count= 1         mutualinfo= 0.000571
----------------------------
583     [this]

count=31        (proba=0.002100)

 --> [price]                count= 15        mutualinfo= 0.000338
----------------------------
588     [ticket]

count=22        (proba=0.001490)

 --> [NUM   ]               count= 535       mutualinfo= 0.000022
 --> [PLACE ]               count= 1416      mutualinfo= -0.000073
----------------------------
589     [tickets]

count=6 (proba=0.000406)

 --> [PLACE ]               count= 1416      mutualinfo= 0.000054
----------------------------
590     [time]

count=19        (proba=0.001287)

 --> [number]               count= 90        mutualinfo= 0.000211
----------------------------
592     [to]

count=686       (proba=0.046464)

 --> [with]                 count= 29        mutualinfo= -0.000029
 --> [by]                   count= 16        mutualinfo= 0.000193
 --> [after]                count= 38        mutualinfo= 0.000509
 --> [for]                  count= 215       mutualinfo= -0.000225
 --> [in]                   count= 123       mutualinfo= 0.000139
 --> [from]                 count= 467       mutualinfo= -0.000301
 --> [i]                    count= 243       mutualinfo= -0.000237
 --> [on]                   count= 377       mutualinfo= -0.000379
 --> [at]                   count= 77        mutualinfo= -0.000114
----------------------------
593     [tomorrow]

count=1 (proba=0.000068)

 --> [the]                  count= 641       mutualinfo= 0.000307
----------------------------
606     [u]

count=1 (proba=0.000068)

 --> [NUM   ]               count= 535       mutualinfo= 0.000324
----------------------------
608     [under]

count=24        (proba=0.001626)

 --> [in]                   count= 123       mutualinfo= 0.000450
----------------------------
615     [us]

count=13        (proba=0.000881)

 --> [NUM   ]               count= 535       mutualinfo= 0.000074
 --> [airlines]             count= 86        mutualinfo= 0.000252
----------------------------
616     [usair]

count=19        (proba=0.001287)
```

78

```
 --> [coach]            count= 73       mutualinfo= 0.000231
 --> [number]           count= 90       mutualinfo= 0.000211
----------------------------
620      [v]
```

```
count=2 (proba=0.000135)
 --> [u]                count= 1        mutualinfo= 0.000870
----------------------------
```

88

```
              wsj_011.bro
              * * * * * * *

----------------------------
1         [NUM   ]

count=261        (proba=0.025261)

 --> [*rrb*]             count= 18       mutualinfo= 0.000790
 --> [that]              count= 92       mutualinfo= -0.000118
 --> [one]               count= 27       mutualinfo= 0.000053
 --> [NUM   ]            count= 261      mutualinfo= -0.000330
----------------------------
6         [DAY   ]

count=2 (proba=0.000194)

 --> [NUM   ]            count= 261      mutualinfo= 0.000417
----------------------------
18        [*]

count=225        (proba=0.021777)

 --> [liquidity]                 count= 4        mutualinfo= 0.000341
 --> [traders]           count= 29       mutualinfo= 0.000064
 --> [he]                count= 28       mutualinfo= 0.000069
 --> [veto]              count= 10       mutualinfo= 0.000213
 --> [*]                 count= 225      mutualinfo= -0.000222
----------------------------
26        [a]

count=185        (proba=0.017906)

 --> [hutchinson]                count= 1        mutualinfo= 0.000562
----------------------------
33        [about]

count=16         (proba=0.001549)

 --> [FREE  ]            count= 0        mutualinfo= Infinity
----------------------------
82        [after]

count=11         (proba=0.001065)

 --> [for]               count= 98       mutualinfo= 0.000316
----------------------------
118       [among]

count=8 (proba=0.000774)

 --> [for]               count= 98       mutualinfo= 0.000360
----------------------------
157       [appropriations]

count=22         (proba=0.002129)

 --> [limitation]                count= 1        mutualinfo= 0.000859
----------------------------
161       [arbitrage]

count=18         (proba=0.001742)

 --> [the]               count= 630      mutualinfo= -0.000013
 --> [form]              count= 2        mutualinfo= 0.000790
----------------------------
163       [arbs]

count=1 (proba=0.000097)

 --> [they]              count= 38       mutualinfo= 0.000783
----------------------------
180       [as]

count=61         (proba=0.005904)

 --> [in]                count= 184      mutualinfo= -0.000012
 --> [because]           count= 16       mutualinfo= 0.000329
----------------------------
190       [association]

count=1 (proba=0.000097)

 --> [mae]               count= 1        mutualinfo= 0.001291
 --> [*lrb*]             count= 15       mutualinfo= 0.000912
----------------------------
192       [at]

count=40         (proba=0.003871)

 --> [for]               count= 98       mutualinfo= 0.000135
----------------------------
225       [banks]

count=16         (proba=0.001549)

 --> [it]                count= 57       mutualinfo= 0.000339
----------------------------
235       [battle]

count=7 (proba=0.000678)

 --> [traditionalists]               count= 1        mutualinfo= 0.001019
----------------------------
263       [bernstein]

count=8 (proba=0.000774)

 --> [performance]               count= 5        mutualinfo= 0.000776
 --> [president]                 count= 44       mutualinfo= 0.000472
----------------------------
269       [big]

count=27         (proba=0.002613)

 --> [institutional]             count= 3        mutualinfo= 0.000677
----------------------------
315       [broker]

count=2 (proba=0.000194)

 --> [inc.]              count= 6        mutualinfo= 0.000944
----------------------------
318       [brooks]

count=1 (proba=0.000097)

 --> [d.]                count= 3        mutualinfo= 0.001137
----------------------------
325       [bush]

count=10         (proba=0.000968)

 --> [it]                count= 57       mutualinfo= 0.000405
 --> [he]                count= 28       mutualinfo= 0.000504
----------------------------
332       [buyers]
```

```
count=9 (proba=0.000871)
 --> [specialist]                 count= 3        mutualinfo= 0.000830
------------------------------
334      [by]
count=53        (proba=0.005130)
 --> [against]          count= 10       mutualinfo= 0.000415
------------------------------
395      [cheetham]
count=1 (proba=0.000097)
 --> [head]             count= 3        mutualinfo= 0.001137
------------------------------
417      [clients]
count=7 (proba=0.000678)
 --> [traders]          count= 29       mutualinfo= 0.000549
------------------------------
419      [clock]
count=1 (proba=0.000097)
 --> [recorder]         count= 1        mutualinfo= 0.001291
------------------------------
428      [collins]
count=2 (proba=0.000194)
 --> [analyst]          count= 2        mutualinfo= 0.001097
------------------------------
434      [commissions]
count=4 (proba=0.000387)
 --> [performance]                count= 5        mutualinfo= 0.000872
------------------------------
439      [companies]
count=8 (proba=0.000774)
 --> [dealers]          count= 3        mutualinfo= 0.000847
 --> [shops]            count= 1        mutualinfo= 0.001000
------------------------------
441      [company]
count=20        (proba=0.001936)
 --> [it]               count= 57       mutualinfo= 0.000308
------------------------------
443      [compared]
count=3 (proba=0.000290)
 --> [making]           count= 6        mutualinfo= 0.000887
------------------------------
454      [computers]
count=6 (proba=0.000581)
 --> [we]               count= 8        mutualinfo= 0.000750
------------------------------
468      [congress]
count=26        (proba=0.002516)
 --> [bush]             count= 10       mutualinfo= 0.000514
 --> [veto]             count= 10       mutualinfo= 0.000514
------------------------------
502      [control]
count=3 (proba=0.000290)
 --> [reap]             count= 1        mutualinfo= 0.001137
------------------------------
511      [corn]
count=3 (proba=0.000290)
 --> [soybeans]         count= 1        mutualinfo= 0.001137
------------------------------
513      [corp]
count=4 (proba=0.000387)
 --> [mac]              count= 1        mutualinfo= 0.001097
 --> [yields]           count= 3        mutualinfo= 0.000944
------------------------------
565      [danzig]
count=1 (proba=0.000097)
 --> [and]              count= 170      mutualinfo= 0.000574
------------------------------
606      [departure]
count=2 (proba=0.000194)
 --> [that]             count= 92       mutualinfo= 0.000562
------------------------------
621      [deterioration]
count=1 (proba=0.000097)
 --> [industry]         count= 5        mutualinfo= 0.001066
------------------------------
650      [discussing]
count=1 (proba=0.000097)
 --> [attempting]                 count= 1        mutualinfo= 0.001291
------------------------------
707      [editor]
count=3 (proba=0.000290)
 --> [bellows]          count= 1        mutualinfo= 0.001137
------------------------------
711      [edwards]
count=2 (proba=0.000194)
 --> [d.]               count= 3        mutualinfo= 0.001040
------------------------------
726      [emasculate]
count=1 (proba=0.000097)
 --> [swallow]          count= 1        mutualinfo= 0.001291
------------------------------
731      [employer]
count=1 (proba=0.000097)
 --> [peabody]          count= 3        mutualinfo= 0.001137
------------------------------
758      [evans]
```

```
count=2 (proba=0.000194)

 --> [mehta]                  count= 2        mutualinfo= 0.001097
----------------------------
771       [examiner]

count=3 (proba=0.000290)

 --> [*]                      count= 225      mutualinfo= 0.000381
----------------------------
777       [exchange]

count=10          (proba=0.000968)

 --> [futures]                count= 22       mutualinfo= 0.001269
----------------------------
797       [expenses]

count=4 (proba=0.000387)

 --> [payments]               count= 14       mutualinfo= 0.000729
----------------------------
844       [fees]

count=13          (proba=0.001258)

 --> [commissions]                    count= 4        mutualinfo= 0.000739
----------------------------
862       [financially]

count=1 (proba=0.000097)

 --> [editorially]                    count= 2        mutualinfo= 0.001194
----------------------------
875       [flirted]

count=1 (proba=0.000097)

 --> [executed]               count= 2        mutualinfo= 0.001194
----------------------------
888       [for]

count=98          (proba=0.009485)

 --> [in]                     count= 184      mutualinfo= -0.000078
 --> [for]                    count= 98       mutualinfo= 0.000010
----------------------------
890       [forces]

count=4 (proba=0.000387)

 --> [guard]                  count= 10       mutualinfo= 0.000776
----------------------------
914       [from]

count=35          (proba=0.003388)

 --> [in]                     count= 184      mutualinfo= 0.000066
 --> [from]                   count= 35       mutualinfo= 0.000298
 --> [but]                    count= 33       mutualinfo= 0.000306
----------------------------
915       [ftc]

count=5 (proba=0.000484)

 --> [department]                     count= 11       mutualinfo= 0.001656
----------------------------
930       [funds]

count=12          (proba=0.001161)

 --> [funds]                  count= 12       mutualinfo= 0.000597
----------------------------
935       [futures]

count=22          (proba=0.002129)

 --> [markets]                count= 19       mutualinfo= 0.000448
----------------------------
952       [gilts]

count=1 (proba=0.000097)

 --> [bonds]                  count= 1        mutualinfo= 0.001291
----------------------------
970       [government]

count=10          (proba=0.000968)

 --> [watchdogs]                      count= 1        mutualinfo= 0.000969
----------------------------
1004      [harder]

count=3 (proba=0.000290)

 --> [harder]                 count= 3        mutualinfo= 0.000984
----------------------------
1013      [he]

count=28          (proba=0.002710)

 --> [he]                     count= 28       mutualinfo= 0.000360
 --> [she]                    count= 4        mutualinfo= 0.000632
----------------------------
1031      [her]

count=3 (proba=0.000290)

 --> [paper]                  count= 13       mutualinfo= 0.000779
----------------------------
1032      [herald]

count=13          (proba=0.001258)

 --> [paper]                  count= 13       mutualinfo= 0.000574
----------------------------
1057      [house]

count=14          (proba=0.001355)

 --> [the]                    count= 630      mutualinfo= 0.000022
----------------------------
1069      [i]

count=9 (proba=0.000871)

 --> [i]                      count= 9        mutualinfo= 0.000677
----------------------------
1071      [if]

count=23          (proba=0.002226)

 --> [in]                     count= 184      mutualinfo= 0.000125
----------------------------
1076      [illinois]

count=1 (proba=0.000097)

 --> [nebraska]               count= 1        mutualinfo= 0.001291
```

```
----------------------------
1089      [in]

count=184         (proba=0.017809)

 --> [if]                 count= 23        mutualinfo= 0.000125
 --> [to]                 count= 285       mutualinfo= -0.000227
 --> [in]                 count= 184       mutualinfo= -0.000166
 --> [on]                 count= 67        mutualinfo= -0.000025
 --> [at]                 count= 40        mutualinfo= 0.000047
 --> [by]                 count= 53        mutualinfo= 0.000008
 --> [from]               count= 35        mutualinfo= 0.000066
 --> [with]               count= 54        mutualinfo= 0.000005
----------------------------
1091      [inc.]

count=6  (proba=0.000581)

 --> [inc]                count= 4         mutualinfo= 0.000847
----------------------------
1093      [included]

count=3  (proba=0.000290)

 --> [are]                count= 47        mutualinfo= 0.000600
----------------------------
1107      [indiana]

count=1  (proba=0.000097)

 --> [illinois]           count= 1         mutualinfo= 0.001291
----------------------------
1109      [indications]

count=1  (proba=0.000097)

 --> [lending]            count= 1         mutualinfo= 0.001291
----------------------------
1143      [into]

count=13          (proba=0.001258)

 --> [with]               count= 54        mutualinfo= 0.000375
----------------------------
1154      [investor]

count=13          (proba=0.001258)

 --> [he]                 count= 28        mutualinfo= 0.000467
 --> [and]                count= 170       mutualinfo= 0.000215
----------------------------
1155      [investors]

count=16          (proba=0.001549)

 --> [swings]             count= 3         mutualinfo= 0.000750
----------------------------
1161      [iowa]

count=3  (proba=0.000290)

 --> [minnesota]                  count= 2         mutualinfo= 0.001040
----------------------------
1166      [it]

count=57          (proba=0.005517)

 --> [congress]           count= 26        mutualinfo= 0.000271
 --> [appropriations]             count= 22        mutualinfo= 0.000294
 --> [it]                 count= 57        mutualinfo= 0.000162

----------------------------
1180      [johnson]

count=1  (proba=0.000097)

 --> [editor]             count= 3         mutualinfo= 0.001137
----------------------------
1193      [keep]

count=4  (proba=0.000387)

 --> [beat]               count= 4         mutualinfo= 0.000903
----------------------------
1194      [keeping]

count=1  (proba=0.000097)

 --> [trying]             count= 4         mutualinfo= 0.001097
----------------------------
1201      [kidder]

count=4  (proba=0.000387)

 --> [unit]               count= 3         mutualinfo= 0.000944
----------------------------
1229      [lawyers]

count=2  (proba=0.000194)

 --> [officials]                  count= 6         mutualinfo= 0.000944
----------------------------
1278      [lobbies]

count=1  (proba=0.000097)

 --> [lawmakers]                  count= 1         mutualinfo= 0.001291
----------------------------
1302      [lotter]

count=1  (proba=0.000097)

 --> [funds]              count= 12        mutualinfo= 0.000944
----------------------------
1309      [lynch]

count=3  (proba=0.000290)

 --> [inc.]               count= 6         mutualinfo= 0.000887
----------------------------
1322      [make]

count=25          (proba=0.002420)

 --> [exceed]             count= 2         mutualinfo= 0.000744
----------------------------
1331      [managers]

count=12          (proba=0.001161)

 --> ['']                 count= 78        mutualinfo= 0.000335
----------------------------
1343      [markets]

count=19          (proba=0.001839)

 --> [themselves]                 count= 3         mutualinfo= 0.000726
----------------------------
1347      [mason]

count=2  (proba=0.000194)
```

```
 --> [chairman]          count= 7        mutualinfo= 0.000922
----------------------------
1355    [mccabe]

count=1 (proba=0.000097)

 --> [officer]           count= 1        mutualinfo= 0.001291
----------------------------
1365    [mehta]

count=2 (proba=0.000194)

 --> [president]                 count= 44       mutualinfo= 0.000665
----------------------------
1400    [money]

count=20        (proba=0.001936)

 --> [FREE   ]           count= 0        mutualinfo= Infinity
----------------------------
1407    [morrison]

count=1 (proba=0.000097)

 --> [olson]             count= 1        mutualinfo= 0.001291
----------------------------
1422    [much]

count=11        (proba=0.001065)

 --> [potentially]               count= 1        mutualinfo= 0.000956
----------------------------
1426    [murray]

count=3 (proba=0.000290)

 --> [securities]                count= 1        mutualinfo= 0.001137
 --> [chairman]          count= 7        mutualinfo= 0.000866
----------------------------
1446    [nebraska]

count=1 (proba=0.000097)

 --> [dakotas]           count= 1        mutualinfo= 0.001291
----------------------------
1456    [neuberger]

count=2 (proba=0.000194)

 --> [berman]            count= 2        mutualinfo= 0.002388
----------------------------
1471    [noble]

count=2 (proba=0.000194)

 --> [media]             count= 1        mutualinfo= 0.001194
----------------------------
1502    [of]

count=272       (proba=0.026326)

 --> [on]                count= 67       mutualinfo= -0.000079
 --> [for]               count= 98       mutualinfo= 0.000063
 --> [after]             count= 11       mutualinfo= 0.000540
 --> [in]                count= 184      mutualinfo= -0.000247
----------------------------
1526    [on]

count=67        (proba=0.006485)

 --> [to]                count= 285      mutualinfo= -0.000086
 --> [on]                count= 67       mutualinfo= 0.000116
 --> [by]                count= 53       mutualinfo= 0.000149
----------------------------
1528    [one]

count=27        (proba=0.002613)

 --> [himself]           count= 1        mutualinfo= 0.000830
----------------------------
1581    [parent]

count=1 (proba=0.000097)

 --> [corp.]             count= 7        mutualinfo= 0.001019
----------------------------
1587    [partisans]

count=1 (proba=0.000097)

 --> [so]                count= 18       mutualinfo= 0.000887
----------------------------
1590    [pasadena]

count=1 (proba=0.000097)

 --> [beach]             count= 2        mutualinfo= 0.001194
----------------------------
1597    [peabody]

count=3 (proba=0.000290)

 --> [lynch]             count= 3        mutualinfo= 0.000984
----------------------------
1656    [power]

count=14        (proba=0.001355)

 --> [appropriations]            count= 22       mutualinfo= 0.000491
----------------------------
1664    [prebon]

count=1 (proba=0.000097)

 --> [u.s.a]             count= 1        mutualinfo= 0.001291
----------------------------
1675    [president]

count=44        (proba=0.004259)

 --> [manager]           count= 3        mutualinfo= 0.000609
----------------------------
1722    [programs]

count=5 (proba=0.000484)

 --> [you]               count= 10       mutualinfo= 0.000744
----------------------------
1747    [public]

count=5 (proba=0.000484)

 --> [they]              count= 38       mutualinfo= 0.000558
----------------------------
1780    [rate]

count=9 (proba=0.000871)

 --> [FREE   ]           count= 0        mutualinfo= Infinity
```

```
------------------------------
1781      [rates]

count=11        (proba=0.001065)

 --> [municipalities]             count= 3        mutualinfo= 0.000802
------------------------------
1855      [reporter]

count=1  (proba=0.000097)

 --> [furillo]             count= 1        mutualinfo= 0.001291
------------------------------
1861      [request]

count=4  (proba=0.000387)

 --> [about]               count= 16       mutualinfo= 0.000710
------------------------------
1903      [riese]

count=3  (proba=0.000290)

 --> [express]             count= 17       mutualinfo= 0.000742
------------------------------
1921      [ruling]

count=5  (proba=0.000484)

 --> [it]                  count= 57       mutualinfo= 0.000501
------------------------------
2000      [shops]

count=1  (proba=0.000097)

 --> [companies]                  count= 8        mutualinfo= 0.001000
------------------------------
2018      [signore]

count=1  (proba=0.000097)

 --> [trader]              count= 3        mutualinfo= 0.001137
------------------------------
2037      [smith]

count=1  (proba=0.000097)

 --> [d.]                  count= 3        mutualinfo= 0.001137
------------------------------
2054      [source]

count=4  (proba=0.000387)

 --> [inc]                 count= 4        mutualinfo= 0.002001
 --> [*lrb*]               count= 15       mutualinfo= 0.000719
------------------------------
2058      [soybeans]

count=1  (proba=0.000097)

 --> [commodities]                count= 3        mutualinfo= 0.001137
------------------------------
2070      [speculators]

count=1  (proba=0.000097)

 --> [risk]                count= 3        mutualinfo= 0.001137
------------------------------
2071      [speed]

count=2  (proba=0.000194)

 --> [movements]                  count= 2        mutualinfo= 0.001097
------------------------------
2087      [staffs]

count=1  (proba=0.000097)

 --> [enforcement]                count= 5        mutualinfo= 0.001066
------------------------------
2095      [stanley]

count=1  (proba=0.000097)

 --> [peabody]             count= 3        mutualinfo= 0.001137
------------------------------
2101      [stay]

count=3  (proba=0.000290)

 --> [work]                count= 3        mutualinfo= 0.000984
------------------------------
2119      [stoll]

count=2  (proba=0.000194)

 --> [authority]                  count= 3        mutualinfo= 0.001040
------------------------------
2127      [street]

count=14        (proba=0.001355)

 --> [and]                 count= 170      mutualinfo= 0.000205
------------------------------
2147      [successor]

count=4  (proba=0.000387)

 --> [dolan]               count= 1        mutualinfo= 0.001097
------------------------------
2174      [sweatshirts]

count=1  (proba=0.000097)

 --> [sparkplugs]                 count= 1        mutualinfo= 0.001291
------------------------------
2190      [tall]

count=1  (proba=0.000097)

 --> [energetic]                  count= 1        mutualinfo= 0.001291
------------------------------
2207      [the]

count=630       (proba=0.060976)

 --> [trading]             count= 51       mutualinfo= -0.000158
------------------------------
2213      [then-speaker]

count=1  (proba=0.000097)

 --> [wright]              count= 1        mutualinfo= 0.001291
------------------------------
2214      [there]

count=9  (proba=0.000871)

 --> [there]               count= 9        mutualinfo= 0.000677
------------------------------
```

```
2216      [they]

count=38          (proba=0.003678)

 --> [they]                 count= 38        mutualinfo= 0.000275
----------------------------
2242      [to]

count=285         (proba=0.027584)

 --> [in]                   count= 184       mutualinfo= -0.000227
 --> [from]                 count= 35        mutualinfo= 0.000005
 --> [for]                  count= 98        mutualinfo= -0.000084
----------------------------
2250      [top]

count=5  (proba=0.000484)

 --> [program-trading]              count= 8        mutualinfo= 0.000776
----------------------------
2262      [trading]

count=51          (proba=0.004936)

 --> [computers]                    count= 6        mutualinfo= 0.000491
 --> [itself]               count= 1         mutualinfo= 0.000742
----------------------------
2280      [trust]

count=2  (proba=0.000194)

 --> [FREE   ]              count= 0         mutualinfo= Infinity
----------------------------
2296      [ual]

count=2  (proba=0.000194)

 --> [shares]               count= 2         mutualinfo= 0.001097
----------------------------
2299      [uncertainty]

count=1  (proba=0.000097)

 --> [deeds]                count= 1         mutualinfo= 0.001291

----------------------------
2319      [unneeded]

count=1  (proba=0.000097)

 --> [even]                 count= 11        mutualinfo= 0.000956
----------------------------
2375      [waited]

count=1  (proba=0.000097)

 --> [sneaked]              count= 1         mutualinfo= 0.001291
----------------------------
2382      [was]

count=25          (proba=0.002420)

 --> [according]                    count= 4        mutualinfo= 0.000648
----------------------------
2412      [widget]

count=7  (proba=0.000678)

 --> [price]                count= 8         mutualinfo= 0.000729
----------------------------
2423      [with]

count=54          (proba=0.005226)

 --> [for]                  count= 98        mutualinfo= 0.000093
----------------------------
2427      [wizards]

count=1  (proba=0.000097)

 --> [clients]              count= 7         mutualinfo= 0.001019
----------------------------
2444      [year]

count=25          (proba=0.002420)

 --> [sales]                count= 8         mutualinfo= 0.000551
----------------------------
```

Annex III

some C programs from the study

```
/*-------------------------------------------
 * NAME      :     global.c
 * CREATED   :    September 27. 1994
 * CHANGED   :    December  1.  1994
 *
 * global variables
 *-------------------------------------------
 */

#include "ldd.h"
#define GLOBAL_C                /*** variables ***/

WORD
 Dico[MAXWORD],
 TabTag[TAGSNB]                 =
 {"xx", "x", "adj", "adjp", "advp", "intj", "np", "ord", "pp", "s", "sbar", "sbarq", "sinv", "sq", "vp",
"whadvp", "whnp", "whpp"};
COUNT
 TabCount[MAXWORD];             /* occurence count r */

TABDEPENDENCE
 BrosDependence,                /*(w1,w2) -> mutual count c */
 SonDependence,
 BigramDependence;
CARDINAL
 WordCard ,                     /* Card {    w      :     count=r} */
 BrosCard ,                     /* Card {(w2 bros w1) :   count=c} */
 SonCard  ,
 BigramCard;
TABLE
 Brothers                       = { "BROTHERS" ,    & BrosDependence ,    & BrosCard    },
 Parents                        = { "PARENTS"  ,    & SonDependence  ,    & SonCard     },
 Bigrams                        = { "BIGRAMS"  ,    & BigramDependence,   & BigramCard};
DEPEND
 Depend                         = { & Brothers, & Parents, & Bigrams };

int;
 WordNb                         =0,
 OccurenceNb                    =0,     /* OccurenceNb >= WordNb */
 SentenceLength                 =2,     /* max length */
 MaxCount                       =MAXCOUNT,
 BigramsRetrouves               =0,
 BigramsNouveaux                =0,
 BrothersRetrouves              =0,
 BrothersNouveaux               =0,
 ParentsRetrouves               =0,
 ParentsNouveaux                =0;
float
 Transfact                      =1;

SENTENCES                       Sentences;
LINEARSENTENCES                 LinearSentences;
FLOATINGDEP                     DataDepSentences [MAXSENTENCE];

int
 NbSentence                     =0,
 Start                          =1,
 Stop                           =MAXSENTENCE,
 Pas                            =1;
```

```
float
 MinInfo              =0.0,
 MinProba             =1.0/10000,
 Lambda1              =0.6,
 Lambda2              =0.5,
 Lambda3              =0.3,
 Lambda4              =0.3;

/*** functions ***/

int
 Index            (WORD, enum FLAGS),
 New_Line         (LINE*, int* __cursor, FILE*);

void
 Text             (WORD, INDEX, enum FLAGS),
 AddinDico        (WORD),
 Open_File        (char*,FILE**);

void
 Init_Transitiv   (TABDEPENDENCE),
 Transfere        (TABDEPENDENCE),
 Write_Tab        (TABDEPENDENCE, FILE*),
 Zero_Liste       (TABDEPENDENCE),

 Display_Xinfo    (TABDEPENDENCE),
 Init_Proba       (TABDEPENDENCE),
 Init_Cardinal_Dico (void),
 Display_Cardinal (CARDINAL),
 Zero_Cardinal    (CARDINAL),
 Init_Cardinal    (TABLE *),
 Zero_Dep         (TABLE *);

int
 Max_Count        (CARDINAL),
 Word_Size        (DEPSENTENCES),
 Show_Set         (DEPSENTENCES);

float
 Mutual_Info      (int __c1,int __c2,int __c1_2),
 XInfo            (TABDEPENDENCE , INDEX);

float
 Bigram_Perplexity (LINEARPHRASE*,int* __length),
 All_Perplexity   (PHRASE*,int* __length,float __Lambda1,float __lambda2, float __Lambda3,float
__Lambda4);

float
 Bigram_Proba_S   (int __index1 ,int __index2),   /*P~(index2 | index1)*/
 Cond_Proba       (int __index1 ,int __index2, TABLE),
 Cond_Proba_Sm    (int __index1, int __index2, TABLE),
 All_Cond_Proba   (int __bigram,int __parent,int __bro,int __word,float __Lambda1,float __Lambda2,
float __Lambda3,float __Lambda4),
 Weight           (int __indWord,int __indBigram,int __indDep, TABLE ,float __lambda),

 Brother_Weight   (WORDDEP, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 Parent_Weight    (WORDDEP, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 Proba            (WORDDEP, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 ProbaBig         (WORDDEP, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 LogProba         (WORDDEP, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 LogProbaBig      (WORDDEP, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 ProbaDep         (WORDDEP, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
```

```
 Perp_Dep           (FLOATINGDEP, int* __Length, float __lambda1,float __lambda2, float
__Lambda3,float __Lambda4),
 Compute_Dep        (FLOATINGDEP, float        (*)(WORDDEP,float,float,float,float), int*
__Length, float __lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 Perplexity_Big     (DEPSENTENCES ),
 Perplexity         (DEPSENTENCES, float __lambda1, float __lambda2, float __Lambda3,float
__Lambda4);

void
 Open_Dico          (STRG),
 Read_Sentence      (STRG),
 Create_Linear      (STRG),
 Display_Tree_L     (PHRASE*),        /*left*/
 Display_Tree       (PHRASE*),        /*right*/

 All_Search         (PHRASE*),

 Search_Bros2       (PHRASE*),
 Search_Son         (PHRASE*),
 Search_Bigram      (LINEARPHRASE *),

 Search_Tri         (FLOATINGDEP *, PHRASE*),

 New_Dep            (FLOATINGDEP,int __pos,int __dep,INDEX __indexDep,INDEX __indexWord),
 Init_Dep           (FLOATINGDEP *),
 Add_Dep            (FLOATINGDEP),
 Display_Dep        (FLOATINGDEP),
 Display_Fonc       (DEPSENTENCES, float(*)(WORDDEP,float,float,float,float),float
__lambda1,float __lambda2, float __Lambda3,float __Lambda4),
 Display_Prob       (FLOATINGDEP,float __lambda1,float __lambda2, float __Lambda3,float
__Lambda4),
 EMConvergence      (DEPSENTENCES, float *__lambda0,float  __epsilon,  int __dep),
 Dichotomy          (DEPSENTENCES, float *__lambda0, float  __epsilon, char __dep);
```

```
/*-----------------------------------------------
 *  NAME :    types.h
 *  CREE :    28 Septembre 94
 *  CHANGE : 19 Octobre 94
 *------------------------------------------------
 */
#ifndef TYPES_H
#define TYPES_H

#define MAXWORD       10000
#define MAXSENTENCE   5000
#define MAXCOUNT      2000  /* max occurence of a bigram */
#define MAXLEN        40
#define LINESIZE      80
#define MAXDEPEND     4   /* sons | brothers | bigrams |trigrams | ...*/


/***words...***/


typedef char          STRG[MAXLEN];
typedef STRG          WORD;
typedef int           COUNT;          /* occurence count */
typedef float         PROBA;
typedef int           INDEX;          /* word index in the dictionary */
typedef char          LINE[LINESIZE];
typedef struct   nextword            /*(word1 --> list of word2s)*/
{
  INDEX               Index2;
  COUNT               Count;         /* count (word1, word2) */
  PROBA               Proba;         /* I(word2 , word1) */
  struct nextword    *Next;
}                     NEXTWORD;
typedef NEXTWORD     *TABDEPENDENCE[MAXWORD];
typedef int           CARDINAL[MAXCOUNT];
typedef struct table
{
  STRG                 Title;
  TABDEPENDENCE        *Liste;
  CARDINAL             *Cardinal;
}                      TABLE;
typedef TABLE          *DEPEND[MAXDEPEND];



/***sentences...***/


enum FLAGS            { FTAG , FWORD };
typedef struct phrase
{
  enum FLAGS           Flag;  /* Key will be either a TAG# or a WORD# */
  int                  Key;
  struct phrase       *Son;
  struct phrase       *Bros;  /* Brothers: Phrases on the same vertical*/
}                      PHRASE;
typedef PHRASE        *SENTENCES[MAXSENTENCE];

#endif
```

1

```
/*----------------------------------------
 *  NAME : index.h
 *  CREE :   Septembre 94
 *  CHANGE : 3 Octobre 94
 *
 *  constant tables = equivalent classes
 *-----------------------------------------
 */

#ifndef INDEX_H
#define INDEX_H

#include "global.h"

#define PLACESNB 16
#define MONTHSNB 11
#define DAYSNB 7
#define ORDISNB 4
#define TAGSNB 15

enum TAGS
{
  X, ADJP, ADVP, INTJ, NP, PP, S, SBAR, SBARQ, SINV, SQ, VP, WHADVP, WHNP, WHPP
  };
WORD        TabTag[TAGSNB] =
{
  "x", "adjp", "advp", "intj", "np", "pp", "s", "sbar", "sbarq", "sinv", "sq", "vp", "
whadvp", "whnp", "whpp"
  };

WORD        TabPlace[PLACESNB] =
{
  "pittsburgh","denver","philadelphia","atlanta","altanta","washington","boston","san"
,"francisco","baltimore","dallas","dc","oakland","texas", "maryland", "stapleton"
  };

/** "may" is not in TabMonth because of its verb homonyme **/
WORD        TabMonth[MONTHSNB] =
{
  "january","february","march","april","june","july","august","september","october","n
ovember","december"
  };

WORD        TabDay[DAYSNB] =
{
  "monday","tuesday","wednesday","thursday","friday","saturday","sunday"
  };

WORD        TabOrdinal[ORDISNB] =
{
  "th","rd","st","nd"
  };

#endif
```

```
/*--------------------------------------------
 *   NAME : ldd.c
 *   CREATED :   September 1994
 *   CHANGED :   Decembre 5 1994
 *
 *    main program of dependencies research
 *--------------------------------------------
 */

#include "global.h"

int select (DEPSENTENCES*,DEPSENTENCES*,DEPSENTENCES,int,int);

DEPSENTENCES
  DepSentences,
  TrainSentences,
  ConvSentences,
  TTestSentences,
  TestSentences;
int
  amount;
float
  epsilon=1E-3,
  mu1,
  mu2;

/*************************************************************/
int select (Set1,Set2,Main,n,m)
     DEPSENTENCES  *Set1, *Set2, Main;
     int n,m;
{
  int i, s1, s2;

  s1 = s2 = i = 0;
  while ( Main[++i])
    if ( (i%m) < n)
      (*Set1)[++s1] = Main[i];
    else
      (*Set2)[++s2] = Main[i];
  return(s1);
}

/*************************************************************/
void Opening(filename)
     STRG filename;
{
  STRG file_par,file_dic;
  int i;
  printf("\n--------OPENING--------\n");
  strcpy(file_par,filename);
  strcat(file_par, ".par");
  strcpy(file_dic,filename);
  strcat(file_dic, ".dic");
  printf("        %s  %s\n",file_par,file_dic);
/****/
  printf("  OPENING DICO \n");
  Open_Dico(file_dic);
  printf("Words :             %d\n", WordNb);
  printf("Words Occurences : %d\n", OccurenceNb);
  Init_Cardinal_Dico ();
  MaxCount = Max_Count(WordCard);
  MinProba = 1.0 / (MaxCount*WordNb) ; /* will be minimum conditionnal proba *
  printf("Max LogProba  ~  %f\n", - log(MinProba) / log(2));
  /***
    for(i=0;i<=WordNb;i++)
    printf("%d\t%s\t%d\n",i,Dico[i],TabCount[i]);
```

```
    /***
      Display_Cardinal  (WordCard);
      /***/
 /*****/
   printf("  CREATING TREE \n");
   Read_Sentence(file_par);
   amount = NbSentence-Start+1;
   printf("Sentences :          %d\n", amount);
   printf("Sentence Length :  %d\n", SentenceLength);
   /***
   printf("  DISPLAYING \n");
   for (i=Start; (i<=Stop)&&(i<=NbSentence); i++)
     {
       printf("\nSentence___%d\n",i);
       Display_Tree(Sentences[i],0);
       printf("\n");
     }
     /***/
}
/***********************************************************/
void Creating()
{
  int i;

  printf("--------CREATING VECTORS--------\n");
  for (i=Start; (i<=Stop)&&(i<=NbSentence); i++)
    {
      DepSentences[i] = & DataDepSentences[i];
      Init_Dep( DepSentences[i] );
      Search_Tri  ( DepSentences[i] , Sentences[i]);
    }
}
/***********************************************************/
void Dividing(n_train, m_train, n_conv, m_conv)
     int n_train, m_train, n_conv, m_conv;
{
  float ratio_train, ratio_conv;
int
  amount_train,amount_dep, amount_conv, amount_test;

  printf("--------DIVIDING DATA--------\n");
  ratio_train = 1.0 * n_train / m_train;
  ratio_conv  = 1.0 * n_conv  / m_conv;
  printf("ratio_train = %f\n",ratio_train);
  printf("ratio_conv  = %f\n",ratio_conv);
  amount_train = select(& TrainSentences, & TTestSentences, DepSentences, n_tra
n, m_train);
  amount_conv  = select(& ConvSentences, & TestSentences, TTestSentences, n_con
, m_conv);
  amount_test  = amount - amount_train - amount_conv ;
  /**/
  printf("%d sentences                \n", amount);
  printf("\t%d sentences for training  \n", amount_train);
  printf("\t%d sentences for converging\n", amount_conv);
  printf("\t%d sentences for testing   \n", amount_test);
  /**
  printf("total     size : %d words\n", Word_Size(DepSentences));
  printf("training  size : %d words\n", Word_Size(TrainSentences));
  printf("converging size : %d words\n", Word_Size(ConvSentences));
  printf("testing   size : %d words\n", Word_Size(TestSentences));
  /**/
}
/***********************************************************/
void LostFound()
{
  printf("found-brothers  %d ",BrothersRetrouves);
```

```
  printf("/ new-brothers  %d\n",BrothersNouveaux);
  printf("found-parents   %d ",ParentsRetrouves);
  printf("/ new-parents   %d\n",ParentsNouveaux);
  printf("found-bigrams   %d ",BigramsRetrouves);
  printf("/ new-bigrams   %d\n",BigramsNouveaux);
}
/************************************************************/
float Testing(Set)
     DEPSENTENCES Set;
{
  float perplexity;

  perplexity = Perplexity_Big(Set);
  printf("PERPLEXITY bigrams  only and not smoothed    : %f\n",perplexity);
  perplexity = Perplexity(Set,0 ,0 ,0 ,0);
  printf("PERPLEXITY bigrams  only                     : %f\n",perplexity);
  perplexity = Perplexity(Set, Lambda1, 0, Lambda1, 0);
  printf("PERPLEXITY bigram / brother                  : %f\n",perplexity);
  perplexity = Perplexity(Set, 0, Lambda2, 0, Lambda2);
  printf("PERPLEXITY bigram / parents                  : %f\n",perplexity);
  perplexity = Perplexity(Set, Lambda1, Lambda2, Lambda3, Lambda4);
  printf("PERPLEXITY brothers + parents + bigrams     : %f\n",perplexity);
  printf("          (%f) (%f) (%f) (%f)\n", Lambda1,Lambda2,Lambda3,Lambda4);
  LostFound();
  return(perplexity);
}


/************************************************************/
float Searching(TrainSentences)
    DEPSENTENCES TrainSentences;
{
  int i=0;
  float perplexity;

  printf("--------SEARCHING DEPENDENCES--------\n");

  for (i=0; i<=2; i++)
    Zero_Dep (Depend[i]);

  while (* TrainSentences[++i])
    Add_Dep (* TrainSentences[i]);

  for (i=0; i<=2; i++)
    {
      Init_Proba(*Depend[i]->Liste);
      Init_Cardinal (Depend[i]);
      /***
        Display_Cardinal  (*Depend[i]->Cardinal);
      /***
        Transfere(*Depend[i]->Liste);
      /***
        printf("\n   XINFO \n");
        Display_Xinfo(*Depend[i]->Liste);
      /***/
    }
  /***
    printf("\n   Brothers Transitivity\n");
    printf("transitivity factor :  ");
    scanf("%f",&Transfact);
    if (Transfact)
    {
    Init_Transitiv(BrosDependence);
    /**
      Transfere(BrosDependence);
      /**
```

```
        }

        /** train-set perplexity **/
    return(Testing(TrainSentences));
  }
/**********************************************************/
float Converging(ConvSentences)
     DEPSENTENCES ConvSentences;
{
  int i, size;
  float perplexity;

  printf("--------CONVERGING--------\n");
  /**
  Dichotomy  (ConvSentences, &mu1, epsilon, 1);
  printf(">mu1      = %f\n",mu1);
  perplexity = Perplexity  (ConvSentences, mu1, 0 );
  Dichotomy  (ConvSentences, &mu2, epsilon, 2);
  printf(">mu2      = %f\n",mu2);
  perplexity = Perplexity  (ConvSentences, 0, mu2 );
  /**/
  EMConvergence  (ConvSentences, &Lambda1, epsilon, 1);
  printf("Brother             >Lambda1 = %f\n",Lambda1);
  EMConvergence  (ConvSentences, &Lambda2, epsilon, 2);
  printf("Parent              >Lambda2 = %f\n",Lambda2);
  Lambda4 = Lambda2;
  Lambda3 = Lambda1;
  /**/
  EMConvergence  (ConvSentences, &Lambda3, epsilon, 3);
  printf("(with Parent (+brother)  >%f)\n",Lambda4);
  printf("      Brother (+parent)  >%f\n",Lambda3);
  /**
  EMConvergence  (ConvSentences, &Lambda4, .1, 4);
  printf("      Parent  (+brother) >%f\n",Lambda4);
  /**
  EMConvergence2 (ConvSentences, &Lambda3, &Lambda4, epsilon);
  printf("Brother&Parent    >Lambda3 = %f , Lambda4 = %f\n",Lambda3,Lambda4);
  /**/
  return(Testing(ConvSentences));
}



/**********************************************************/



void main(argc,argv)  /**file. **/
  int   argc;
  char  *argv[];
{
  STRG file_par, file_dic;
  int i=0;
  int n_train, m_train, n_conv, m_conv;
  float max_train = .8;
  int on1, on2;
  float Sperp, Cperp, Tperp, Bigperp;
  float perp, nperp;
  printf("\nbegin*********** %s *************\n",argv[1]);

  perp = nperp = Sperp = Cperp = Tperp = 1.0 * MaxCount;
```

```
    Opening(argv[1]);
    Creating();
    n_train = 4;  m_train = 5;
    n_conv  = 1;  m_conv  = 2;
    Dividing(n_train, m_train, n_conv, m_conv);
    Sperp = Searching(TrainSentences);
    /**/
    Stop = 5;
    printf("\n\n 100 * Proba | Big+Par\n");
    Display_Fonc(TrainSentences,Proba,0,Lambda2,0,Lambda4);
    Stop = NbSentence;
    /**/
    Cperp = Converging(ConvSentences);
    printf("--------TESTING--------\n");
    Tperp = Testing(TestSentences);

/***/
   printf("\n******* M_TRAIN\n");
   on1 = 1;
   while (on1)
   {
     Dividing(n_train, m_train, n_conv, m_conv);
     Sperp = Searching(TrainSentences);
     Cperp = Converging(ConvSentences);
     printf("--------TESTING--------\n");
     Tperp = Testing(TestSentences);
     if ( (nperp - Tperp>= 0) && (n_train/m_train < max_train) )
       {
         m_train = m_train + 2;
         n_train++;
         nperp = Tperp;
       }
     else
       {
         on1 = 0;
         m_train = m_train - 2;
         n_train--;
         Tperp = nperp;
       }
     printf("\n*******\n");
   }
/***/
  printf("\n******* M_CONV\n");
  m_conv++;
  on2 = 1;
  while (on2)
    {
      Dividing(n_train, m_train, n_conv, m_conv);
      Sperp = Searching(TrainSentences);
      Cperp = Converging(ConvSentences);
      printf("--------TESTING--------\n");
      Tperp = Testing(TestSentences);
      if ( (nperp - Tperp>= 0) && (n_train/m_train < max_train) )
        {
          m_conv++;
          nperp = Tperp;
        }
      else
        {
          on2 = 0;
          m_conv--;
          Tperp = nperp;
        }
      printf("\n*******\n");
    }
  printf("\n*******\n");
```

```
 /**
 printf("\nBrother >Lambda1 = %f\n",Lambda1);
 printf("Parent  >Lambda2 = %f\n",Lambda2);
 printf("Brother&Parent    >%f , %f\n",Lambda3,Lambda4);
 /**
 Bigperp = Perplexity_Big(TestSentences);
 printf("\nbigrams only                  : %f\n",Bigperp);
 printf("parents + brothers + bigrams : %f\n",Tperp);
 LostFound();
 /**
 printf("\nN_TRAIN %d\t/   M_TRAIN %d\n", n_train, m_train);
 printf("N_CONV  1 \t/   M_CONV  %d\n", m_conv);
 /**
 printf("Train Set:\n%d\n",Show_Set(TrainSentences));
 printf("Test  Set:\n%d\n",Show_Set(TestSentences));
 /**
 Stop = 8;
 printf("\n 100 * Proba | Bigram\n");
 Display_Fonc(TestSentences, ProbaBig,0,0,0,0);
 /**
 printf("\n\n 100 * Proba | All\n");
 Display_Fonc(TestSentences, Proba, Lambda1, Lambda2, Lambda3, Lambda4);
 /**
 printf("\n\n 100 * Proba | Brother\n");
 Display_Fonc(TestSentences, ProbaDep, 1, 0, 1, 0);
 printf("\n\n 100 * Proba | Parent\n");
 Display_Fonc(TestSentences, ProbaDep, 0, 1, 0, 1);
 Stop = NbSentence;
 /**
 printf("\n");
 while(*TestSentences[++i] && i < 5)
 Display_Dep ( *TestSentences[i] );
 /**/
 printf("\n***************** %s **************end\n",argv[1]);
}
```

```
#
# Makefile for ldd -     September 28. 1994
#                        November 15. 1994
#

EXTHDRS      =    /usr/include/ansi_compat.h \
                  /usr/include/stdio.h \
                  /usr/include/stdlib.h \
                  /usr/include/math.h \
                  /usr/include/string.h

CC           =    gcc

CFLAGS       =    -c -g

LINKER       =    gcc

LDFLAGS      =    -g

LIBS         =    DEFS =       -DDEBUG

DEST         =    ${HOME}/bin

MAKEFILE     =    Makefile

HDRS         =    types.h \
                  ldd.h \
                  global.h \
                  opendico.h \
                  readsentence.h \
                  search.h \
                  index.h

SRCS         =    ldd.c \
                  global.c \
                  opendico.c \
                  readsentence.c \
                  createlinear.c \
                  displaytree.c \
                  allsearch.c \
                  searchbigram.c \
                  allperplexity.c \
                  newsearch.c \
                  floatingdep.c \
                  convergence.c \
                  conditionnalproba.c \
                  mutualinfo.c \
                  xinfo.c \
                  initcounts.c \
                  addcouple.c \
                  lookflag.c \
                  index.c \
                  openfile.c \
                  read.c
```

```
OBJS            =   ldd.o \
                    global.o \
                    opendico.o \
                    readsentence.o \
                    createlinear.o \
                    displaytree.o \
                    allsearch.o \
                    searchbigram.o \
                    allperplexity.o \
                    newsearch.o \
                    floatingdep.o \
                    convergence.o \
                    conditionnalproba.o \
                    mutualinfo.o \
                    xinfo.o \
                    initcounts.o \
                    addcouple.o \
                    lookflag.o \
                    index.o \
                    openfile.o \
                    read.o

PROGRAM         =   ldd

PRINT           =   pr

all:                $(PROGRAM)

.c.o:
                    $(CC) $(CFLAGS) $(DEFS) $<

$(PROGRAM):         $(OBJS) $(LIBS)
                    $(LINKER) $(LDFLAGS) $(OBJS) $(LIBS) -lm -o
$(PROGRAM)

clean:              rm -f $(OBJS)

install:            $(PROGRAM)
                    install -s $(PROGRAM) $(DEST)
program:            $(PROGRAM)

update:             $(DEST)/$(PROGRAM)

$(DEST)/$(PROGRAM): $(SRCS) $(LIBS) $(HDRS) $(EXTHDRS)
                    @make -f $(MAKEFILE) DEST=$(DEST) install

###
```