Information theoretic approaches to computational linguistics

John Goldsmith

version 2019-7
Chapters 1 - 5

September 7, 2023

Contents

1 Organization

My primary office is in Rosenwald 201B, and I can be found there. It is in the Linguistics Department. I'm available a lot of the time; please email ahead and we can fix a time. For help with coding, it is best that you work first with one of the TAs.

There are two TAs for the undergraduate version of this course. They are grad students in Computer Science: Steven Basart (xdsteven@uchicago.edu) and Lang Yu (langyu@uchicago.edu).

The graduate version of this course is for graduate students in Computer Science, and they will have an extra project to do—a team project on discovering compounds and multiword units (MWEs).

Students will submit their homeworks through the CS department's SVN server. There is information on how to use it on the chalk website for this course.

There are 9 regular problem sets (1 for each week, though the graduating students will automatically get full credit for the last problem), and full score on all problems gives 90 points, which qualifies as an A. To facilitate your normal expectations about numbers, I will add 10 free points to everyone's score so that full score looks like 100 rather than 90.

Problem sets begin to be due in Week 2.

There will be no hourly exams or final; the grade is based on your assignments, plus class participation. The HMM problem is the hardest, I think, and you really will have to set aside the time needed for it during the middle three weeks of the quarter.

1.1 Homework assignments

Your assignment should in general contain the following components:

1. A README.txt file

It should begin with a table explaining the nature of the different files you have submitted. Here is an example from a student's homework last year:

Contains:	
anagrams.py	-Run this. -Usage: python anagrams.py [-i] -F -N where -F is followed by the filename -N is followed by the min size of the anagram set The [-i] flag will cause the program to print out the most interesting anagram (Problem 6). Without the flag, the program sorts the anagrams by size, length, and overlap. Sample usage: python anagrams.py [-i] -Fdict.txt -N6
formatdata.py	-Module to parse data
stringDistance.py	-Module from hw6 to calculate string edit distance.
dict.txt	-file to run the above on
sampleOutput.txt	-output from running the above on dict.txt and size 6 (as requested by the assignment and as given by the sample usage above)
sampleSurprisingOutput.txt	-output from running the above on dict.txt and size 1

Then you should explain in prose what you have done.

2. Typically, you will have a file with a name such as "output.txt", whose significance will be evident to the human eye, aided by the explanations you include.

1.2 Data

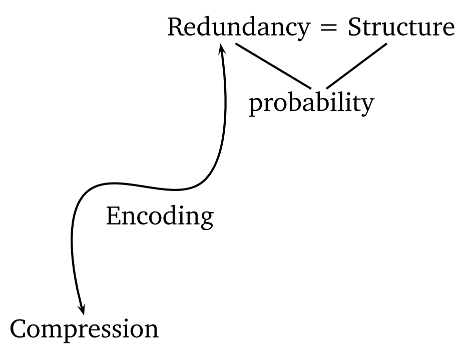
I have made a set of dx1 files from previous years available at `www.cs.uchicago.edu/~jagoldsm/data`.

Introduction

2

2.1 What we are going to do in this course

This course has evolved over the past few years into a course that has a specific orientation: it is a course about a number of methods that are widely used in computational linguistics (and related fields), and it is a course about how quantitative methods and models can be used to gain considerable insight into the structures of natural (human) languages. We could do the first without worrying about the second, and it would still be a course on natural language processing and computational linguistics, but it would be much less interesting, and not worthy of the University of Chicago in any event. To date, most use of quantitative models in modern linguistics have been add-ons, after the fact—efforts to see how quantitative data can be used to support, clarify, or infirm hypotheses that arise out of other methods. That is not what we will be doing. Our goal is to find ways in which quantitative methods can be developed in order to deepen our understanding of what grammatical structure is (using the term 'grammatical structure' in the broadest way possible).



Observation

= Structure + Message + Noise

Structure ⇒ Prediction

2.1.1 Kinds of models

- Transitional probability models: FSAs
- Segmentation models
- Vector space models

Anagrams pairs	
licenses	silences
algorithm	logarithm
cautioned	education
continued	unnoticed
generates	teenagers
grandiose	organised
integrals	triangles
percussion	supersonic
striptease	tapestries
colonialist	oscillation
entirety	eternity

Tab. 2.1: Anagram pairs

2.1.2 Anagrams

What's the best way to write a program to find anagrams?

I took a list of 44,000 English words, and found 1,200 sets of anagrams. Most of them are boring, but how do we characterize 'boring' (so we can go beyond knowing it when we see it)?

Some interesting ones are in table 2.1 (see just below).

Others are almost interesting, like *countrimen* and *unromantic* (who spells it *countrimen*?), or *coordinates* and *decorations*, or *incorporate* and *procreation*.

A lot are boring for reasons that are easy to make explicit: there are many, many examples of the form *brother's*, *brothers'*. What is the principle that makes those cases uninteresting?

There are quite a few which are misspellings, like *provdied* and *provided*, or *available* and *avaliable*. That's not interesting either. Why?

Likewise, *takeover* and *overtake*, or *nine-thirty* and *thirty-nine*.

Or even *conversation* and *conservation*.

How about these:

I think we can all agree that last one is really boring, and so is the second to last. But the first one is quite striking – worth remembering, to amaze people at a party (but not to tell them what you learned at school today).

Anagrams				
alerting	altering	integral	relating	triangle
enlist	inlets	listen	silent	tinsel
least	slate	stale	steal	tales
post	pots	spot	stop	tops

Tab. 2.2: Anagrams 2

There are two points to this brief look at anagrams. The first is the answer to the question, what's the right strategy for writing a program to identify anagrams? That's a good question, with a clear answer. The second question is interesting, but much vaguer – still, some of you might find it interesting. It is: is it possible to write a program that will model our intutitions of what an interesting set of anagrams is, as opposed to boring anagram sets? What contributes to some anagram sets being boring? Surely length has a lot to do with it (*post* and *pots*, e.g.) but also semantically unusual and surprising juxtapositions (like *integral* and *triangle*: it is relevant that they both sit in the semantic field of mathematics, and yet otherwise have nothing to do with each other, at least as far as letters are concerned–or *listen* and *silent*).

As to the first question—how to find anagrams—the point is that this is a case where we want to do a little bit of preprocessing of the data, and it will make the job vastly more simple. We don't want to directly compare integral and triangle and compare the letters one by one. We want to find a representation for each word that eliminates the linear ordering that makes each word what it is. And often the most natural way to eliminate a certain kind of information in a representation (here, the linear ordering of the letters) is to impose a regular pattern. The regular pattern that we will impose is this: we will begin by alphabetizing the letters of each word. Then we can sort (that is, alphabetize) those alphabetized lists, and all anagrams will end up next to each other on that list. Couldn't be easier.

2.2 State of the discipline of computational linguistics; its relationship to neighboring domains.

[1]

This is the first, last, and only class that address intellectual history as such. And I can offer right now the most important thing I will say all quarter: **You don't understand an answer until you understand the question to which it is the answer.** And we —we teachers—tend to teach answers more often than we teach questions. And we test how well you understand the answers, not how well you understand the questions. Sometimes we ourselves don't understand the questions to which we are offering the answers.

[1] Week 1, class 2

The second most important thing I can say is that **the work of each generation is the challenge posed by two things: first, the technological advances offered to it and the advances made in other fields, and second, a response to the perceived inadequacies of the answers offered by their parents' generation.** This second is very important: each generation is very sensitive and aware of the inadequacies of what is handed to it by its teachers' generation. Ironically, of course, the teachers' generation passed on what it thought was most important, part of which was the corrections that it offered of their teachers' generation —and the irony then emerges for natural reasons that each generation tends to feel a kinship to their grandparents' (intellectual grandparents) generation.

When we look at the way in which disciplines change from a generational point of view, it is helpful to separate *internal* or *endogenous* change, such as the desire of one generation to separate itself and its problems from those of the generation before, from the *external* or em exogenous changes, notably political events (such as World War II, or wars in general) and changes in technology (which we as computer scientists are very aware of).

Remember: the long tradition of viewing the goal of scientific knowledge as *understanding, prediction, and control*. "Understanding" is intimately connected to the notion of "explanation," a term which means different things in different contexts (and times).

2.3 Brief history of computation, linguistics and computational linguistics

Reading: Perreira, Abney (twice), Lillian Lee. Easy but very thoughtful papers.

2.3.1 History of modern computation

Began with Leibniz and Pascal, and their concern with the difference between soft mentalism and hard mentalism. At this time, the principal observable difference was that people can think, and no on else can. Some people can thinker better than others; is that just like the observation that some people are stronger than others? Is being able to think more a kind of strength, or mental strength?

By the 17th century, thinkers were becoming more concerned about *certainty*. The model for certainty was Descartes (soft logic) and Euclid (hard logic). Mathematics was also a model for knowledge; things were true or false, known or unknown. Things known through hard logic could be accessible to a machine. Machines were new at this point. Is this a paradox, that mechanical things can know the same things as are the most certain? Maybe so.

This became a more important question in the mid 19th century (George Boole, Gottlob Frege) especially with David Hilbert. Part of the concern was the apparent destruction of the foundations of certainty, which was held up to the light with the changing status of geometry. (Kant, non-Euclidian geometry)

George Boole tried to show that logical inference could be reduced to algebra. But at the same time he accepted the (perhaps minority view) that knowledge must be divided into what is known with certainty and what is known with probability.

David Hilbert in the late 1880s provided a new axiomatization for Euclidian geometry.

What would the foundations for arithmetic be? There was a lot of buzz and excitement about the possibility of reducing arithmetic to set theory and logic. Peano, Frege, Russell.

Then Russell discovered his antinomy, and set theory no longer seemed the best place to build the foundations of arithmetic.

Hilbert wanted to build a theory of mathematical inference, and that was done by several people in the 1920s. Turing, Church, Godel.

Turing's approach (and Post's) involved an abstract object that looked very much like machines that we have in the Real World.

Concern in the 1920s and 1930s about the notion of an effective procedure. Alan Turing and his Turing machine (a-machine). World War II, and the three projects of the Allies that needed a computer: artillery aim, code-breaking, atomic bomb modeling.

After World War II, there were serous concerns that the country would fall back into a depression. No one knew whether the advent of computers would help or hurt the economy in this way.

2.3.2 History of [mainstream] linguistics

19th century linguistics focused on history—history of European languages, and the reconstruction of Proto-Indo-European, and its sense was that explanation meant historical explanation. Reasons based on desire to understand the origin of peoples (nations), genealogically and geographically.

Discovery of Indo-European, with some uncertainty as to where it was spoken. Still a live question today.

Linguistics began in Germany, and rose with the *rise of the University* (Wilhelm von Humboldt). The research university was a new invention. Students come in order to learn how to do research and make new discoveries. They are not learning to become preachers, or bureaucrats, or ministers, or high school teachers. This system was strong in Germany, and not so much elsewhere in Europe. It came to the US with Johns Hopkins, Clark, and the University of Chicago. 4 generations of American universities: Harvard, land-grant, Humboldtian, cold-war.

Great interest in where we come from, biologically and culturally (and hence linguistically). Proto-IndoEuropean. Relation to philology.

Shift around 1900, with Saussure, a shift to interest in synchronic linguistics.

In the 20th century, three new modes of explanation arose for the study of language: **psychological, sociological**, and **algorithmic**. Chomsky's principal contribution was bringing algorithmic thinking, and a sense that such analysis provided a new kind of explanation, to mainstream linguistics. This was part of a larger movement that took place in the 1950s, in large part an effect of the rise of computers and computational modes of thinking.

Edward Sapir and Leonard Bloomfield were the two greatest American linguists of the first half of the 20th century; they were followed by Zellig Harris and Charles Hockett, both informally students of both Sapir and Bloomfield. Chomsky was the student of Harris.

1924: founding of Linguistics Society of America. American structuralist tradition: Bloomfield, Sapir, Zellig Harris, Charles Hockett. Anti-metaphysical orientation, in keeping with the strong positivist trend of the time. Zellig Harris: the goal of the linguist is to produce an account of how and where a language departs from equiprobability of its component pieces.

Cycles of views about abstract objects and positivism.

Excursus on George Zipf, and his laws. Opinions at the time about quantitative generalizations in linguistic analysis. Zipf's Law: inverse linear relation between frequency and frequency rank. Zipf perceived a hostility towards anything at all quantitative among the LSA linguists. Dynamic philology.

2.3.2.1 Formal linguistics

Development of the notion of grammar as a generalization of the formalization of proof, starting in the 1930s, notably through the work of the Polish logicians, e.g. Lukasiewicz. Konnexität.

1940s: development of information theory: Norbert Wiener, Claude Shannon (and Warren Weaver of the Rockefeller Foundation).

1950s: Chomsky and Solomonoff: problem of induction, and the possible relevance of probability theory.

Computers: embraced by many linguists in 1950s and 1960s. More, perhaps, in England even than in the US.

2.3.2.2 Back to computers

Computers 1940s, heavily supported by WWII (solving diff equations on the fly for artillery; decoding; development of nuclear weapons)

Shannon, cybernetics, machine translations (MT) as part of the Cold War effort.

Cybernetics: Norbert Wiener. Machine translation: Warren Weaver of the Rockefeller Foundation wrote a famous letter in c. 1948. Cold war, support from CIA, starting in 1951: MIT, Johns Hopkins especially; about a dozen different places, mostly universities.

Not a great success. Oversold project, expectations were too high. Bar-Hillel. Started project at MIT; wrote a report in the early 1960s that killed MT in the US.

Information theory: late 1940s: Shannon, Wiener, Weaver and Shannon. Wiener's best seller!

Complexity: Ray Solomonoff, Greg Chaitin, Kolmogoroff. Solomonoff: University of Chicago and Carnap. Cambridge MA.

2.3.2.3 Computational linguistics

MT:Journal named *Mechanical Translation* founded in 1954 by Victor Yngve (U Chicago). Name changed to: *Mechanical Translation and Computational Linguistics*, and taken over by ACL in 1965.

ACL founded 1962, then called *AMTCL*; became *ACL* in 1968.

Two killer apps:

1. Machine translation: MT
2. ASR, automatic speech recognition.

[30m] 1990: rise of statistical models, probabilistic models, esp. in speech recognition. ASR. ASR was the second thing on everyone's wish list. Air Force was a buyer for this; they want pilots be able to speak to their airplanes. Medical demand. Brief digression on speech generation; interest in the 1990s. Email before smart phones. In the late 1980s, a group began to split off, these were Jelinek/IBM model, and Jim Brown at Dragon Systems; the HMM view;

In this era, the task is not to model experts, but to create systems that learn when you give them a lot of examples of what they have to learn: supervised learning. This seemed like way to high a goal, but it is the perspective that has won the day.

Second huge methodological change: insistence on quantitative evaluation of performace. Big change in the values of what counts as computational linguistics.

The problem is not to write a program that can recognize what you're saying; the problem is write a program that can take a large corpus, tagged as needed, and itself develop a capacity to make the link between sound and spelling.

Mainstream linguistics does not share these values! Why not? Why not learning + quantitative evaluation.

Answer: Rise of generative grammar in the late 1950s. At first it seemed to many that Chomsky would bring a link to computaitonal linguistics, but Chomsky himself always said that that was a misunderstanding of what he was trying to do. He thinks that the problem of learning grammar from data is unrealistic; the reason that we as children do it is that we have a good of prior knowledge, a rich learning mechanism by virtue of our genetic endowment. There is no general learning algorithm involved in learning language.

[41m] It would be false to say that all linguists follow him in this, but many do. This is changing, because of big data on the internet; also because there are people like me and classes like this one.

Abney's paper admirably brings out the three themes that statistical computational linguistics brought in the 1990s:

1. a demand for analyses that were robust (did not operate on toy data samples);

2. that were language-independent, and thus learned and were portable;

3. and analyses that were accompanied by quantitative measures of how well they did, or did not, do.

2.3.3 Conflicts

Pereira's paper is excellent, but casting the theme as Chomsky versus Zellig Harris is just a little bit too local (to Penn), IMHO.

The falling out between expert-systems computational linguists and statistical machine-learning computational linguists. This conflict is often symbolized by something Fred Jelinek said (and wrote): **Whenever I fire a linguist our system performance improves.** From "Applying information theoretic methods: evaluation of grammar quality." Workshop on Evaluation of NLP Systems, Dec 1988. [Google on that].

They don't tell you that IBM Research was in the throes of a struggle between the two natural language groups, Jelinek in one (developing applications of HMMs) and the other with George Heidorn and Karen Jensen, who moved to MSR in 1991.

Jelinek: "My colleagues and I always hoped that linguistics will eventually allow us to strike gold. . . The quote accentuated a certain situation that existed in ASR in the seventies and in NLP in the eighties."[2]

Based on Jelinek (2004, talk at JHU):

> "**The situation in the 1970s**:
>
> 1. Rules and AI govern NLP and speech research
> 2. No distinction between training and test
> 3. IBM linguists had respect for but underestimated ASR problem
> 4. Chomsky thought that statistics were illegitimate
> 5. ARPA project on ASR (1971 to 1976) dominated by AI (except for Jim Baker at CMU)
>
> **The View of the IBM Group**
>
> 6. Linguistic intuition combined with ability to extract information will determine the structure of models and their parameterization
> 7. Parameter values will be estimated from (annotated) data
> 8. We will rely on advice of linguists to create resources
> 9. The problem is not of direct interest to linguists
>
> **Creation of Linguistic Resources**
>
> 10. Brown Corpus (1967)
> 11. Lancaster – Oslo- Bergen corpus (1970)
> 12. Lancaster POS tagging by rule (1982)
> 13. Lancaster treebank (1983 -1986): Geoff Leech and Geoff Sampson
> 14. IBM commissions 2 – 3 M word treebank at Lancaster (1987)
> 15. Linguistic Data Consortium, early 1990s
> 16. Penn parser, NSF support starting in 1990.
> 17. ACL 1990: 39 papers, 1 statistical; 2003: 62 papers, 48 statistical.
> 18. Statistical MT at IBM, starting in 1986; DARPA funding in 1991, including IBM (Candide project), Pangloss at CMU, ISI, NMSU) and Lingstat at Dragon (Baker)

[2] from slides on the internet

2.3.4 Chomsky and the chomskian view about language learnability

What is the relative size of the information needed to describe a language, on the one hand, and that needed to describe the learning algorithm? Chomsky believes that the learning algorithm is much larger, and has a lot of essentially arbitrary information that cannot be inferred.

2.3.5 Data! Data! Data!

Poverty of the Stimulus

"There is no data like more data." (Robert Mercer at Arden House, 1985)[3]

"More data is more important than better algorithms." Eric Brill

Annotated or raw?

2.3.6 Relationship today between computational linguistics and mainstream linguistics

Changing quite a bit, but not all that fast.

2.3.7 What is to be optimized

Classical generative grammar:

1. Function $F_1(g[rammar], d[ata]) \rightarrow \{Yes, No\}$

$\hat{g} = argmin_{g s.t. F_1(g,d)=Yes} length(g)$

Find a grammar-language in which length is defined so that the preceding equation is true. This assumes we have an independent and empirical way of characterizing F_1.

2. Modern computational linguistics

For a given grammar model $\mathcal{G}$, find a learning model $\mathcal{L}$ such that we can find $\hat{g} = argmax_g p(g, d)$.

[3]Last November 2017, it was revealed that Robert Mercer had been pressured to step down as co-chief executive because of his support for Breitbart News. Mercer's daughter Rebeka was a major backer of the Trump campaign, and introduced Steve Bannon to the Trump team. She also created the Defeat Crooked Hillary PAC. Mercer was a lead researcher at the IBM group that developed statistical machine translation in the late 1980s and early 1990s.

p(g,d) is not the only expression we could imagine in that expression, but it is a natural place-holder.

3. Minimum description length

Find $\hat{g} = argmin_{g \in \mathcal{G} = algorithms?}(length(g) + plog_g(d))$

$= argmax(2^{-length(g)} \times p_g(d))$

2.4 Final note

By the 1970s, the **Chomskian wing of linguistic theory** had come to one positive and one negative conclusion that were relevant to the interests of computational linguists: (1) *Language is not just one thing after another* (one phoneme after another, one letter after another, one word after another): it is highly structured, and the discovery of that structure is the interesting part of studying language. (2) *It is not possible that this "hidden" structure is learned, so it must be innate.*

By the mid 1980s or a bit later perhaps, the mainstream of computational linguistics had come to a complementary and divergent perspective: many language problems that had seemed completely intractable could be dealt with much better if *a probabilistic model was employed, and these models were very naturally understood as models of language learning*. The models were partly structural (they had inherent structure, built by the scientist/engineer) but they had many parameters that were empirically trained.

Probabilistic models for speech recognition: Let us just consider the case where individual (isolated) words are to be recognized from speech. How do we go from the speech signal to the underlying sequence of phonemes constituting the word?

Speech can be sampled every 10 msec., and speech transcribed into a rich alphabet of 1000 different speech samplets ("codebook")—microphonetic distribution, but then a 0.5 seconds is 50 symbols long. We create a probabilistic model for each word (given its phonemic pronunciation) and build a probability distribution over all sequences from the microphonetic symbols.

Then, given a speech sample S, we chose for the word which assigns the maximum probability to that sample:

Perceived-word = arg $max_{candidate-word}$ pr(S | candidate-word)

Not quite. That's an oversimplification, because its logic derives from Bayes's Rule, and that allows us to more properly infer:

Perceived-word = arg $max_{candidate-word}$ pr(S | candidate-word) pr(candidate-word)

Hidden Markov models.

And then in the early 1990s, the IBM group took this HMM-inspired model and turned it on the problem of machine translation.

2.5 The emphasis on quantitative evaluation

I noted earlier that one of the big innovations in computational linguistics has been its emphasis on quantitative evaluation. (Alas, it has also become an obsession—the field has in some respects gone too far—but that is not a tragedy; it can be fixed over time.) The most familiar method of evaluation is based on precision and recall, and to use these terms, it is necessary to be able to describe what the task is in terms of identifying a discrete set of objects. The first use of these terms arose in the context of document retrieval. If you submit a request for all documents in a system that bear on "finite state automata" by entering that phrase into your system, the system will return a set of documents.

The documents retrieved can be divided into those that should have been retrieved, and those that should not have. Those that were retrieved and indeed should have been are the *true positives*, while those that were retrieved but should not have been are the *false positives*. In addition, we are considering a situation in which we know the full set of documents that we had hoped to have returned to us by our request: that would be the *true* set of relevant documents, and the full set of documents that was returned (i.e., the truly relevant ones) are the *positives*.

Precision and recall are two simple ratios based on these notions. Precision is defined as the ratio of (count of the) true positives to the (count of the) positives, and recall is defined as the ratio of the (count of the) true positives to the (count of the) true set.

Precision $= \frac{\#true\,positives}{\#positives}$;

Recall $= \frac{\#true\,positives}{\#true}$;

But behind those simple statements are hidden secrets. We can take a project and devise several different ways to apply these formulas, and get extremely different numerical results, depending on what it is that we say we are trying to retrieve.

Here is a real example that illustrates this. We are interested in evaluating an algorithm that performs word-breaking: it takes a large corpus in which the spaces have been removed, and it must reverse engineer the corpus in order to discover the words.

What is it we are trying to recover? There are three natural ways to answer that question:

1. Find points between letters where a space should be placed. We will test how many of the spaces put in were correct, and how many correct spaces were missed.

2. Find word tokens in the output (broken) corpus. For example, if the string "theblackcat" is turned into "theblack cat", then it has come up with two words, "theblack" and "cat", of which only the second is correct. If another line of the corpus has that phrase and gets the same analysis, the counts are increased, because we are counting tokens rather than types.

3. Find word types, that is, what proportion of the vocabulary that generated the corpus is actually discovered by the algorithm

2.6 Topics to come

Here are the topics we will be focusing on in this course.

1. Introduction
2. History
3. Resources, etc.
4. Distributions, probability, plots, letters, words, etc.
5. Plogs, extensive and intensive quantities, MI and PMI;
6. KL divergence
7. Memoizing algorithms: words on a page for latex; string edit distance;
8. HM1: FSAs, probabilistic FSA, Viterbi (max) generation
9. HM2:
10. Compression, numeric compression and graphing L to R, R to L compressed form.
11. Words: Sequitur
12. MDL: As a method to compute hypothesis preference, and as a stopping criterion
13. Words: de Marcken (MDL approach), the good and the bad.

14. Word-internal structure: morphology. Initial heuristic based on ZHarris.

15. Morphology: signatures. Using MDL for hypothesis preference.

16. Morphology: problem cases.

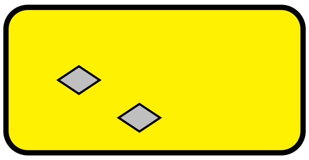
17. Using Linguistica 4, 5.

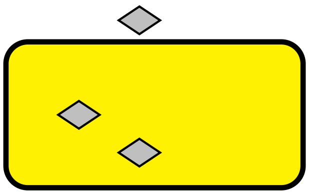
18. Graphs and their uses. Gephi.

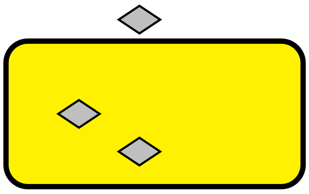
19. Computing word-similarity by context overlaps. Spectral decomposition.

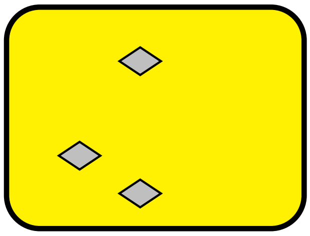
20. Interpreting the graph of word-context similarity. Clustering and social networking.

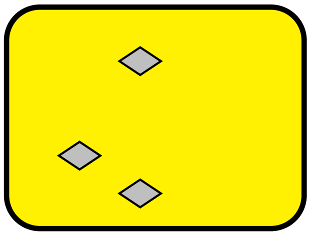
21. Neural nets; word2vec.

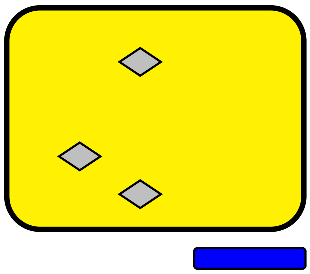


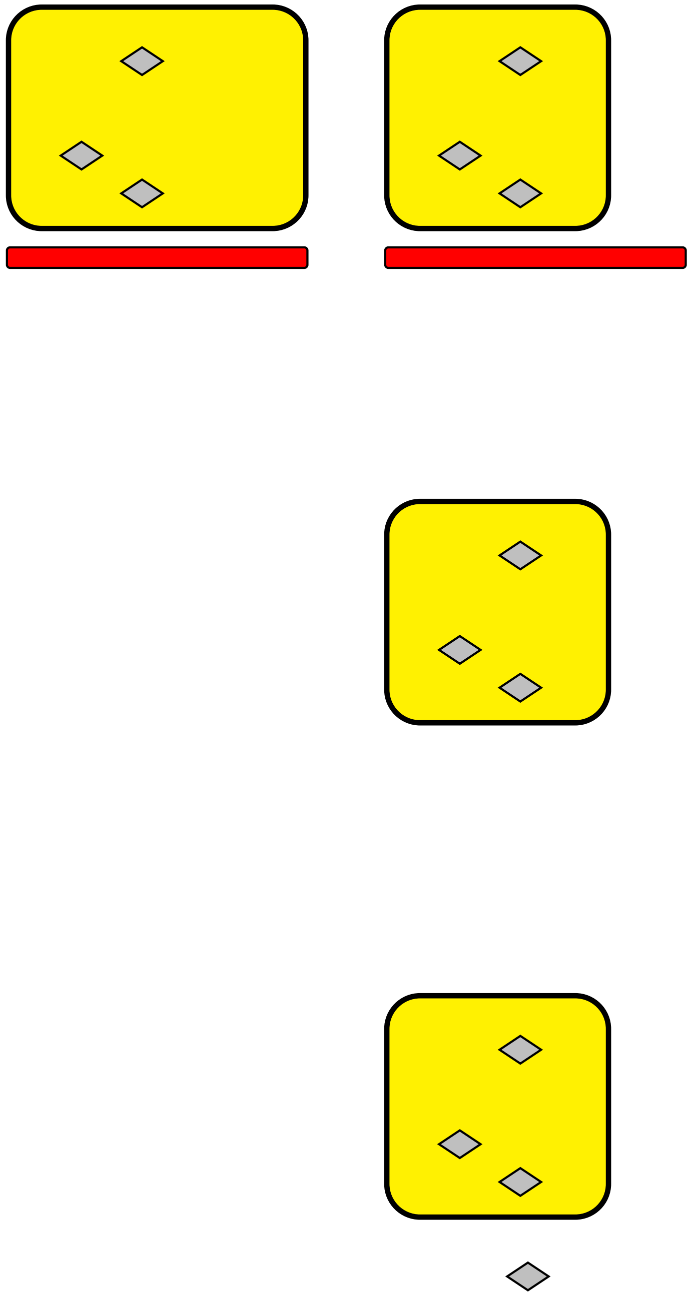


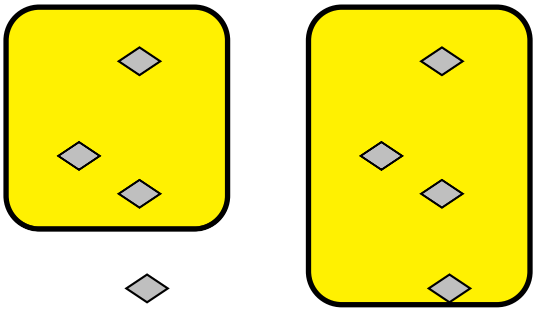


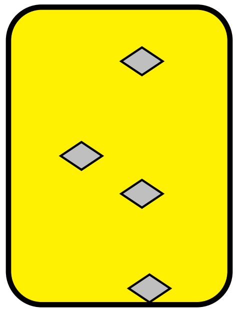


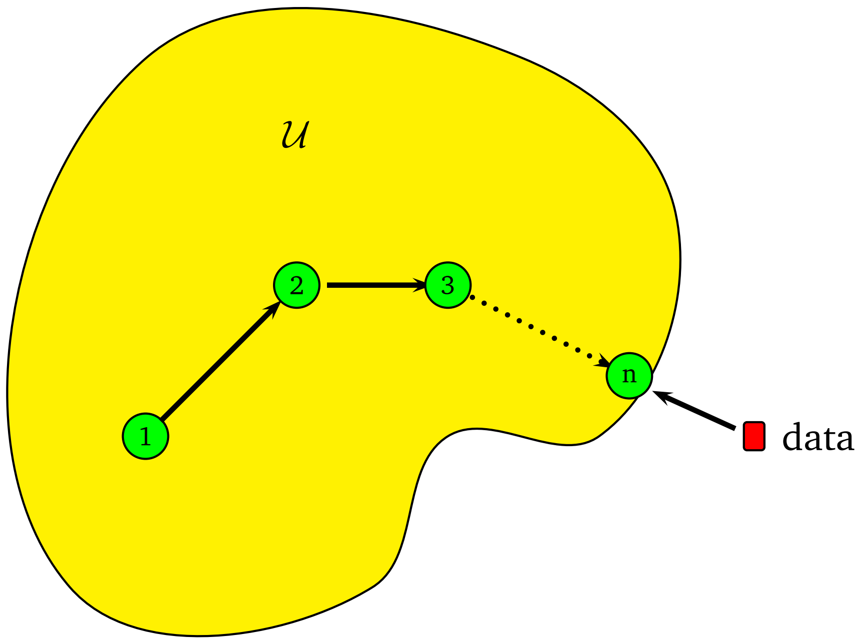











$\mathcal{U}$
2
3
1
n
data

3 Basics of probability and information theory

1. Notation: *max, min, argmax, argmin*. If X={-2, -1, 0, 1, 2} and Y is the real numbers and $f(x) = x^2$, then $max_{x \in X} = 2$ and $argmax_{x \in X} f(x) = 2$. Clear?[1] If it is not clear, it's probably because you are thinking $max_{x \in X} f(x) = 4$, which is true.

2. Probability as the quantitative theory of evidence.

3. Probability distribution: definition.

4. We typically require ourselves to assign a probability distribution over some set, typically infinite, and typically the entire set of strings generated by an alphabet or lexicon.

5. This is a methodological commitment, not a substantive commitment.

6. Difference from the mainstream linguistic assumption that the goal is to create a grammar that generates all and onlGy the well-formed expressions of a language.[2] Importance for NLP to deal with all expected inputs. Grammar checkers.

7. The purpose of assigning probability to data is to test the grammar, not the data.

8. That means we have to think intelligently about the difference between frequency and probability.

9. AOTBE, the best grammar is the one that assigns the highest probability to the grammar.

10. Counts versus frequency (or relative frequency).

11. Good and bad aspects of using observed frequencies as probabilities. Brittleness.

12. Language identification as an NLP project.

13. Difference between letters and words: with letters, they are from an alphabet and you observe the whole alphabet pretty quickly. New words all the time.

14. Explore letters and phoneme frequencies in a couple of languages.

[1] Well, I had written a number other than 2 there. How could that have happened?
[2] Alternative: form-meaning connection.

15. Letter frequencies in corpora and in dictionaries.

16. Zipf.

17. distribution over [0,1] corresponding to first symbol of alphabet (a, or 1); sums to 1.0.

18. probability of a string S = product of probability of each letter * probability of a string of length |S|.

19. nested intervals as a visualization of numeric compression.

20. Conditional probability. Examples: (1) Conditioned on how long a string is.

Probability and the recognition process
By William S. Cooper[3]

An assumption which underlies most of the research being conducted by the MIT Mechanical Translation group is that a mechanical translation process should make a distinction between language recognition and language generation. That is, it is assumed that all input text should be subject to a recognition process which reduces it to what are terms specificers and after suitable manipulation these specifiers should be used to guide the generation of a target-language output. Now, specifiers are defined and labelled by the linguists themselves, so there need be no problem of ambiguity in interpreting the specifiers for the purpose of generating the target-langauge output. However, the recognition process must interpret the source-language input, and so it must meet head-on all the ambiguities with which the source-langauge is fraught. Or more exactly, it must resolve all those ambiguities whose resolutions would affect the choice of specifiers. Since many ambiguities can occur on various levels with a small amount of text, the multiple-choice problem associated with each ambiguity may be aggravated by the ambiguities surrounding it. In this manner, multiple-choice problems can proliferate in a way which makes the task of resolution quite formidable. This paper defends the application of probabilistic methods as an integral part of the recognition process, both as a means of making the best guesses about unresolvable ambiguities and as a technique for speeding up the decision when the ambiguities are resolvable....

By a “multiple choice” we mean situations in which some linguistic unit (e.g. a word) is ambiguous with respect to some finite set of classes. Such a situation would presumably arise because a dictionary or a table has failed to tag the unit with a unique class name at some earlier stage of the process... Indeed, the multiple-choice situation seems to be characteristic of many fields such as speech recognition, character recognition, and library searching, as well as the field presently under discussion. For operations such as our sentence-

[3] Dear Professor Goldsmith, I don’t think I wrote that, or at least I can’t remember doing so, but it is so close to my interests at the time it could almost have been written by me. In 1958 and 1959 I wrote a M.Sc. thesis at MIT on how to use Markov transition probabilities to help resolve ambiguities in automatic translation, or if not resolve them at least to rank the possibilities in order of likelihood. Conceivably this was written as a description of my work, perhaps as part of a grant application or report. William S. Cooper. 4/21/11.

parsing procedure above, there seems to be no recourse from turning to the contexqt for the reduction of ambiguities.

...[F]or both unresolvable and resolvable ambiguities, a probability distribution over the alternatives of a multiple-choice situation would be very useful. For unresolvable ambiguities, it would provide some basis for choice among the alternatives, and would make possible the establishment of a threshold for the elimination of low-probability alternatives. For resolvable ambiguities, probability distributions would permit the ordering of tentative selections, so that an advanced resolution process would never be wasted on an unlikely selection unless the likelier ones had been tried first. In other words, for both kinds of ambiguity, a rough-and-ready technique is needed to assign meaningful probabilities to the alternatives of a multiple choice situation.

For the data necessary to assign the probabilities, we will need two kinds of statistics. The first we will call "dictionary information" and the second, "transition information." The dictionary information is simply a listing of all the possible linguistic units, together with a probability distribution for each unit over the set of classes according to which the units must be tagged. For example, the parsing machine ...would require a dictionary listing each possible word, together with the probability that that word (considered out of context) might be a noun, a pronoun, a verb, etc.. The transition information must be gleaned from a large body of text; it tells the likelihood of a linguistic unit's belinging to a given class, when the class membership of one or more of the preceeding units is known. In our example, we would require the probability that an article, say, would be followed by a noun, a pronoun, etc.. Although these two kinds of information by no means describe completely the probabilistic picture, they are adequate for purposes of obtaining rough guesses about the true class memberships of the linguistic units.

Language data and its models. Alphabet: Σ. Includes # = ' '. All strings: $\Sigma^*; \Sigma^+$. These sets always countable. A subset of Σ^* is a lexicon or a vocabulary. [4]

A corpus or text is a single string in Σ^* that ends with # and contains no internal ##. If it contains at least one internal #, it is 'broken' (a good thing, not bad). It has word-breaks, boundaries.

In most contexts, when we refer to a *word*, we mean a string that ends in # and has no internal #s, like *dog#*.

A broken corpus gives us a lexicon in a natural way.

Notation: $s \in \Sigma^*$; 'dog' S[1] = 'd'; 'dog#' t[1] = 'd'.

[4] Σ^*, lexicon, vocabulary

Elements are discrete: we can assign a non-negative probability to each point.

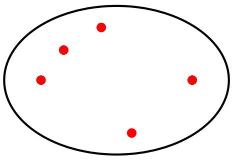


Fig. 3.1: A view of a discrete sample space universe

3.1 Counts, distributions, frequencies, letters, phonemes, probabilities, sequences, words – and finite-state automata (FSAs).

[5]

(Virtually) all our models are discrete for now (later we will have probabilities over models with continuous real-valued parameters). [6] *Notation*: $Count[s] = Count_C[s] = [s]_C = [s]$.

A probability space is a triple: (i) a sample space Ω, the set of all possible outcomes: this could be single draws from a deck; it could a lexicon; it could be a corpus; (ii) a set of *events* or *observable events*, where such an event is a set of outcomes (the phrase "observable event" is sometimes used because it helps make clear the sense that sometimes the underlying events are too 'fine' to be directly observed—a particular real value could be a member of the events, but it is not observable—so we restrict the assignment of probabilities to intervals and sets composed out of intervals); and (iii) a function (*p()*, perhaps)which maps events to reals in [0,1] —and it must be true that $p(\Omega) = 1$. When the sample space Ω is countable, we can assign a probability to each member, and the probability of an event is the sum of the probabilities of the member outcomes.

When it is not discrete, then we are required to limit the sets about which we can ask the question: what probability mass is assigned to them? In particular, we cannot ask what the probability mass is of a point, or a discrete set of points, in a continuous case. All this is the way that probability is generally defined and described, and sometimes it feels like this terminology is not terribly helpful in the cases that we look at, at this point.

[5] Week 2, Monday
[6] This section revised a bit, 10 Jan 2014.

Elements are not discrete: we cannot assign a non-negative probability to each point.

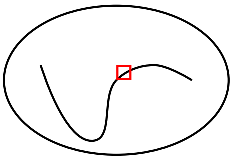


Fig. 3.2: A view of a continuous sample space universe

Map from our sample space to some set of interest.

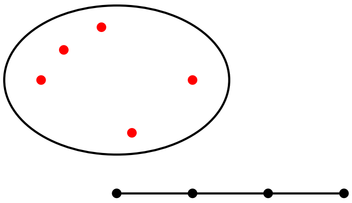


Fig. 3.3: A random variable

When it is discrete (which is the normal case for us), then we can talk about all of the values that the probability function takes on for the possible outcomes, and we call this a *distribution*[7]. A distribution is just a multiset of numbers that sum to 1.0—perhaps in the limit, if the set is infinite. We will be using this term all the time; we will say, for example, that we need to be sure that some multiset of numbers does indeed form a distribution. For this to be true, each number must be greater than or equal to 0, and they must sum (possibly in the limit) to 1.0.

A random variable[8] is a function from the **sample space** Ω to another (measurable) space, called the **state space**. A traditional example is a random variable that maps from coin tosses to a sample space, the set $\{0, 1\}$ (Heads corresponds, let's say, to 0, and Tails to 1.) Each of the two outcomes, 0 and 1, has a probability assigned to it, and those probabilities sum to 1. (In specific contexts, like work on language, we may allow the state space to be words, for example, rather than numbers. And then the words have a probability assigned to them.)

If you read standard introductions to probability, you will find that they focus on numbers and quantity, and there is a good deal of discussion of the cases where the values of the random variable are numbers, and then we look at (for example) the subset of Ω which is the "preimage" of the interval in the state space from $-\infty$ to any particular value x. But even though that is standard, it is really not very helpful, more often than not, in the cases that we look at in this course: we are usually not looking at a range of numerical values between this value and that

[7] **distribution**
[8] Random variables: neither random nor variables.

value. Familiar numerical values are cases where the sample space is a population of people, and the random variable is age. Or the random variable might be age. Both map to real numbers. Both have meaningful means (averages). (And we might look for a relationship between the two random variables: maybe if someone is older, he or she is older.) But when we look at words, for example, it is not generally meaningful to talk about the average of a set of words, or the average of the alphabet of a language.

In cases that we will look at, most sample spaces are either finite alphabets or finite lexicons (strings of symbols from the alphabet), or else they are infinite sequences of elements from an alphabet or lexicon. (Don't take that as a promise that we will not consider cases outside of those cases; it's just a heads-up from a tour-guide.) As I have noted, we are especially interested in the infinite sequences case, and we can either deal with those cases by establishing that the sample space consists of such infinite sequences, *or* we can say that the sample space is the finite alphabet (resp. lexicon), and that we are considering a sequence (possibly infinite) of random variables. [Picture] We sometimes say that these random variables are *indexed* (i.e., by their position in the sequence); we also talk about them comprising a stochastic process.

Example 1. The sample space is the two rolls of a die, (x, y), and the random variable f maps each point to a value $x + y$. The inverse image of 2, $f^{-1}(2)$, is (1,1), and its probability is $\frac{1}{36}$ for a fair die. In this case, we have built the sequentiality—the time element—into the sample space.

Example 2. The sample space is letters of the alphabet, and the random variable is the first letter of each word in this document—which is to say, it is a variable that takes on values in the sample space, and the probability of the inverse image of an element in the sample space (e.g., m) is the probability that the letter m occurs here as the first letter of a word.

Example 3. The sample space is the letters of the alphabet, and we have a sequence of random variables, $X_1, X_2, \ldots, X_5$, which are the first, second, and third letter of a (random) word in this document. In fact, the probability of the values taken on by X_2 is different depending on the value taken on by X_1.

In this case, we talk of a *sequence of random variables*, and what we mean by that is that each point (call it x) in the sample space is mapped by each of the random variables to an element in the range of the random variable.

Most of the time, this kind of modeling is based on the belief that there is an underlying reality that is changing over time, and that time can be modeled discretely, and that reality's shifts can be viewed as shifts from one point in the sample space to another, and that our observations correspond to the values of a random variable, and that the complexity of the evolution of the underlying system can be at least partially understood if we consider the dependencies between random variables X_n and X_{n+i}, where i is a small number (like 1 or 2).

Example 4. Word length studies. (Based on Peter Grzybek 2006)

Thomas Corwin Mendenhall (1841-1924). American physicist and metereologist: purely empirical study, but he looked at the distribution, and not (merely) the average word length.

Sir William P. Elderton (1877-1962). 1949: Geometrical distribution of syllables, based on 1-initial geometrical distribution. P(n) $= p(1-p)^{n-1}$, where n is the number of syllables. He measured a mean length of 1.3487 syllables per word, hence is (its reciprocal) 0.7415.

Syllable count	count	frequency	predicted count	pred freq
1	2987	0.7613	3883	0.7415
2	831	0.1587	1004	0.1917
3	281	0.0537	259.5	0.0496
4	121	0.0231	67	0.0128
5	15	0.0029	17	0.0033
6	2	0.0004	4.48	0.0009

Sergei Chevanov 1897-1955: 1947. Many languages. Also syllable based, but used Poisson distribution: P(n) $= e^{-\lambda}\frac{a^{n-1}}{(n-1)!}$, where a is a free parameter (so-called "1-displaced Poisson distribution").

Does it make sense to study word-length in terms of number of letters/phonemes?

What if we know the average length of a syllable in a particular language?

Is there going to be a relationship between the number of distinct phonemes in a language and the average length of a word?

A geometric distribution will always give the greatest probability to the shorter strings. But a Poisson distribution will not.

3.1.1 Frequencies and probabilities

Some of the basic terms we need to be clear on:

rank	position in a sorted list
count	number of occurrences
frequency	proportion of number of occurrences of *this* divided by *all*
log frequency	
plog frequency	-1 $\times$ log frequency
probability	a value in a distribution
plog [probability]	-1 $\times$ log probability

We can use frequencies as our probabilities, but bear in mind that these two concepts are quite different.

Some additional remarks, spring 2018.

The passage from frequencies to probabilities can be very confusing, and the worst part is that virtually everything we read adds to that confusion; there is a great deal of comfort and ease that comes from ignoring the difference. Let's try to do a little bit better.

The first obstacle concerns the status of *causality.* There are two subparts to this obstacle. First of all, we all share a certain kind of implicit positivism, by which I mean the view that the fundamentally correct way to view the universe is as a complex object in 3 dimensions, whose evolution in time is based on an arbitrarily small window of access to the past. If we are thinking about falling objects, the positivist will remind us that if we know the force of gravity, and the position and the velocity of something falling, then we will be sure of its position and velocity at all moments before it hits the ground. Furthermore, we have a belief that we can account for those facts (location, velocity) in terms of the underlying forces (a gravitational field). The object falls at a certain rate because of its weight and the details of the gravitational field at the various points that the object passes through.

What does this have to do with linguistics and probability? The point is this: We do not need to embrace a belief that the information (= values of parameters, values of a random variable) that we use as conditions are causally related to the outcomes whose probability we wish to estimate.

Two important cases: bigram model conditioning on the left (= past) and conditioning on the right (= future).

We will often consider a model of English or some other language in which the choice of a word (for example) at a moment (or at a point in a string) is conditioned by the previous word. If we do that consistently for a string [including its finality-marking punctuation #)], then if we multiply all of the conditional probabilities, we get a probability for the whole string.

But. . . we get the same probability for the string if we compute its probability from right to left, using a conditional probability for each word based on the word that follows.

So, which is it? Is the sentence generated from left to right, or from right to left? The answer (of course!) is *neither*. We have created two different models which have different usefulnesses, but which agree on a number of calculations of probabilities.

To make matters even more confusing (if that were possible), we can easily imagine a third way to assign a probability to a sentence, one which is more familiar to linguists. We could create a grammar that assigns probabilities to trees rather than words (or words sequences). S becomes NP VP with probability 1.0. NP becomes pronoun (.5), noun (.2), adjective + noun (.3). pronoun becomes he (.3) or she(.4), him (.2), her (.1); noun becomes boy (.4) girl (.5) dog (.1). VP becomes V + NP (.3) or V (.7). V becomes sees (.5) or knows (.5)

We could then calculate the probability of "he sees her" and of "he sees she". Let's do it...

Is this a model of causality? No. Is it a better probabilistic model? Let's calculate and see.

Let[9] us start[10] by looking at a probabilistic model for strings of symbols. The symbols will represent phonemes or letters, but they could also correspond to feature bundles, autosegmental representations, etc. We begin with a finite set of symbols, A, referred to as the *alphabet*. The notation A^+ denotes all strings (sequences) of one or more symbols drawn from A. We have a special symbol in A, #, to represent the word boundary, or space.[11] We then define a *word* as any finite sequence of one or more symbols that ends with $\#$. Given this definition, a *word-set* S is a subset of the set of all possible words: $S \subseteq (A^+\#)$. Similarly, a *word-list* or *corpus* C is an element of the set of all possible sequences of words, $C \in (A^+\#)^*$. The definitions that we give in this section will be for word-sets.

One of the simplest questions that can be asked about a set of words is how often any given single symbol, or *unigram*,[12] appears. For a unigram a, we will write $Count(a)$ to indicate the total number of times that a occurs in all the words in the set. For each symbol $a \in A \cup \{\#\}$, the unigram model induced from a word-set S assigns a probability to a that represents its frequency in the word-set. That is:

$$p(a) = \frac{Count(a)}{|S|} \tag{3.1}$$

where $|S|$ equals the total number of symbols in all the words in S.

For a word $w \in S$, we use the notation $w[n]$ to refer to the n-th symbol in the string (i.e. $w[1]$ is the first symbol, $w[2]$ the second, and so on). Given a word w, the *unigram probability* of w, denoted $p_1(w)$, is defined as the product of the probabilities of the segments comprising the word. For a set of words S, the product of the probabilities of the words is denoted $p_1(S)$. These are given in (3.2a) and (3.2b):

$$a.\;\; p_1(w) = \prod_{i=1}^{|w|} p_1(w[i]) \qquad b.\;\; p_1(S) = \prod_{w \in S} p(w) \tag{3.2}$$

where $|w|$ denotes the number of symbols in w (i.e., the length of the word).

There is a small point here that we must not lose sight of, and it is the main reason we defined a word as a sequence that ends with # (and has no internal #). If we have a distribution over the letters of an alphabet, then the sum of the probabilities of all sequences of length 1 must sum to 1.0. Likewise (though the reasoning takes one extra step algebraically), if we consider the set of all 2-letter strings, and assign each 2-letter string a probability equal to the product of the

[9] Week 2, class 2; Jan 15 2014

[10] I'm using some text from Goldsmith and Riggle *ITAP*

[11] This allows us to assess average word length and to refer to segments at word edges as being adjacent to # in the same way that they are adjacent to their segmental neighbors. Like the symbols for phonemes in A, the symbol # is associated with a probability and can condition the probability of its neighbors. Thus, in what follows, we will refer to # as a phoneme (though it is, in many ways, a different kind of abstract object than a consonant or a vowel).

[12] unigram

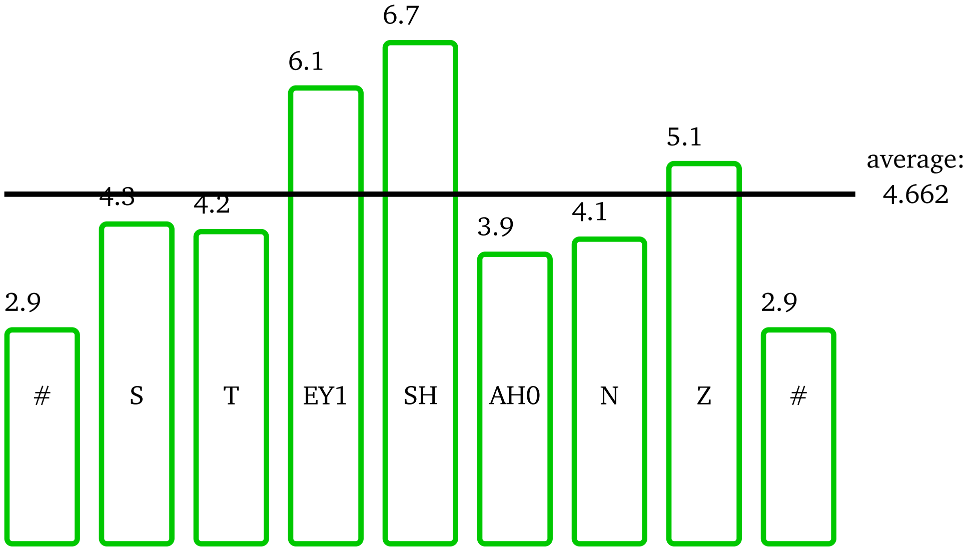


Fig. 3.4: Unigram model

probabilities of its letters, then the sum of those probabilities will also be 1.0. Hence, we can't use this apportioning of probability and expect it to give us a distribution over strings of several different lengths (let alone over all strings of any length). In order to get a distribution over strings of many lengths, we must construct explicitly a distribution over lengths (call it "$\lambda(n)$", perhaps), and then we can say that the probability of a string of 2 letters is the product of the probability of each letter *times* $\lambda(2)$. Or else we can do that implicitly, by setting up a symbol (such as #) which is assigned some of the probability mass that would otherwise go to the letters of the alphabet, and insist that a word ends with #. Can you see that this gives us the right result? It may not be a completely realistic model of word-length, however.

In many cases, the probability computed by a model is the product of a number of distinct factors; because $log(x \times y) = \log(x) + \log(y)$ we can interpret the probability assigned to a form as the sum of the logarithms of these factors. Since $\log(x)$ is negative for $0 < x < 1$, the logs of probabilities are often multiplied by -1 to yield what is referred to as *inverse log probability*; we propose a simpler neologism, the *positive log probability*, or *plog*,[13] for short. Thus (3.2) can be recast with plogs as in (3.3).

$$a.\ \ plog(w) = -\sum_{i=1}^{|w|} \log\ p(w[i]) \qquad b.\ \ plog(S) = -\sum_{w \in S} \log\ p(w) \tag{3.3}$$

The average plog of a word w or word-set S can be calculated as in (3.4a) or (3.4b).

$$a.\ \ -\frac{1}{|w|}\sum_{i=1}^{|w|} \log\ p(w_{[i]}) \qquad b.\ \ -\frac{1}{|S|}\sum_{w \in S} \log\ p(w) \tag{3.4}$$

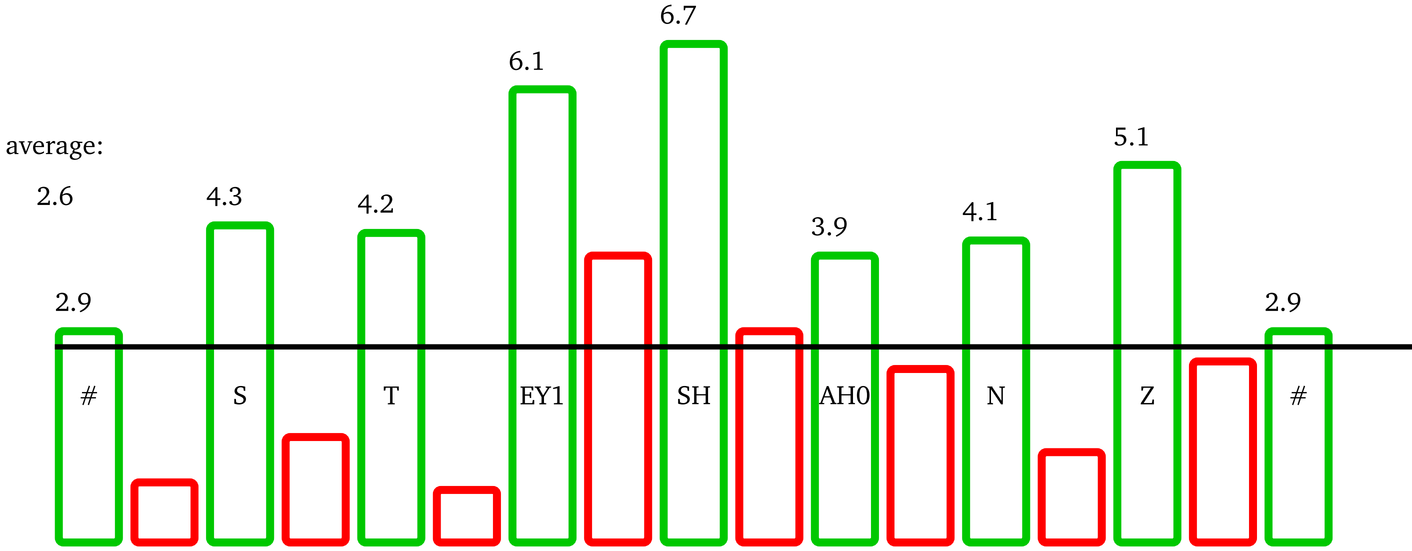
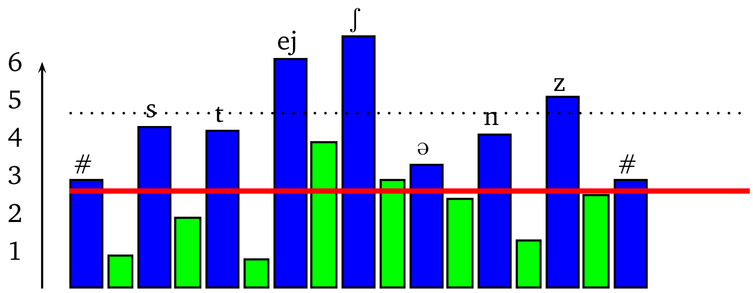


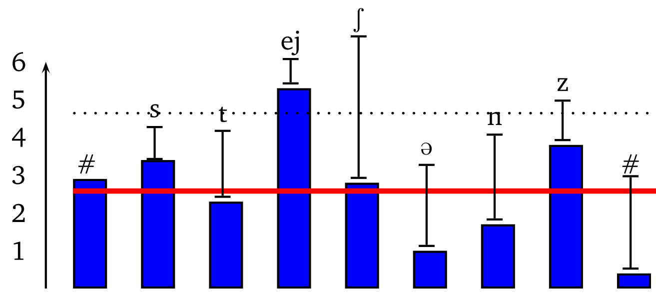
Fig. 3.5: Unigram model with mutual information

The average plog, as calculated in (3.4a), encodes the average complexity[14] of the phonemes comprising the word. If we calculate this figure for all the words of our vocabulary and sort them in light of this figure, the words with the smallest value will be the words largely composed of high frequency phonemes, and the words with the largest values will be words composed largely of low frequency phonemes.

Average below is 2.58 (down from 4.64)



Green: Mutual information in *stations*
Blue: Unigram plot in *stations*



Blue: Log conditional (bigram) probability in *stations*
Decrease from unigram model is exactly the mutual inform

In Table 3.2 we illustrate the range of average plogs from the top ten and the bottom ten of a sample of 63,204 English words along with the positive logs of the frequencies of the top and bottom ten of 54 English phonemes. The data combines a modified version of the CMU English lexicon weighted by word frequencies based on counts from the Brown corpus. The particular transcriptions that appear may raise some eyebrows, but we have used their transcription throughout, though we have used here American phonetic symbols rather than the Darpabet.

[13] plog
[14] Not obvious, perhaps, that we want to use the word that way. We will talk about this.

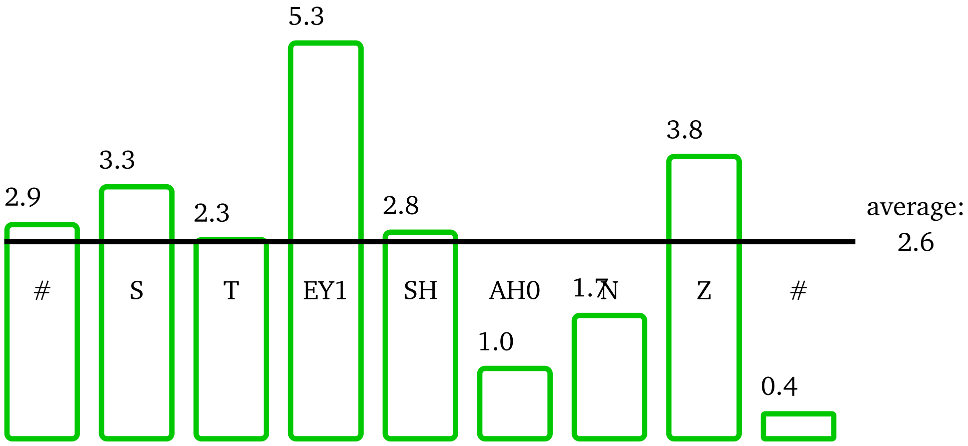


Fig. 3.6: Bigram model

rank	orthography	phonemes	*avg. plog*
1	a	ə	3.11
2	an	ən	3.44
3	to	tə	3.47
4	and	ənd	3.80
5	eh	ɛ́	3.88
63,200	geoid	ǰíɔ̆yd	7.40
63,201	Cesare	čĕzárĕ	7.40
63,202	Thurgood	θɚ́gʌ̆d	7.47
63,203	Chenoweth	čɛ́nɔ̆wɛ̆θ	7.49
63,204	Qureshey	kəréšĕ	7.54

Tab. 3.1: Top and bottom five words and phonemes by average *plog*

If one takes the (not uncontroversial) position that markedness is correlated with frequency, then the plogs in this table would be seen as roughly quantitative estimates of various segments' markedness.[15]

Tab. 3.2: Top and bottom five words and phonemes by average *plog*

[15] In constraint-based models other than Optimality Theory [?] that allow violability but eschew strict domination such as, e.g., Pater, Potts, and Bhatt ([?]), Hayes and Wilson ([?]), or [?] one could view the unigram model as setting up a constraint against each segment, and weighting the violation of constraint $*a$ by the value $plog(a)$.

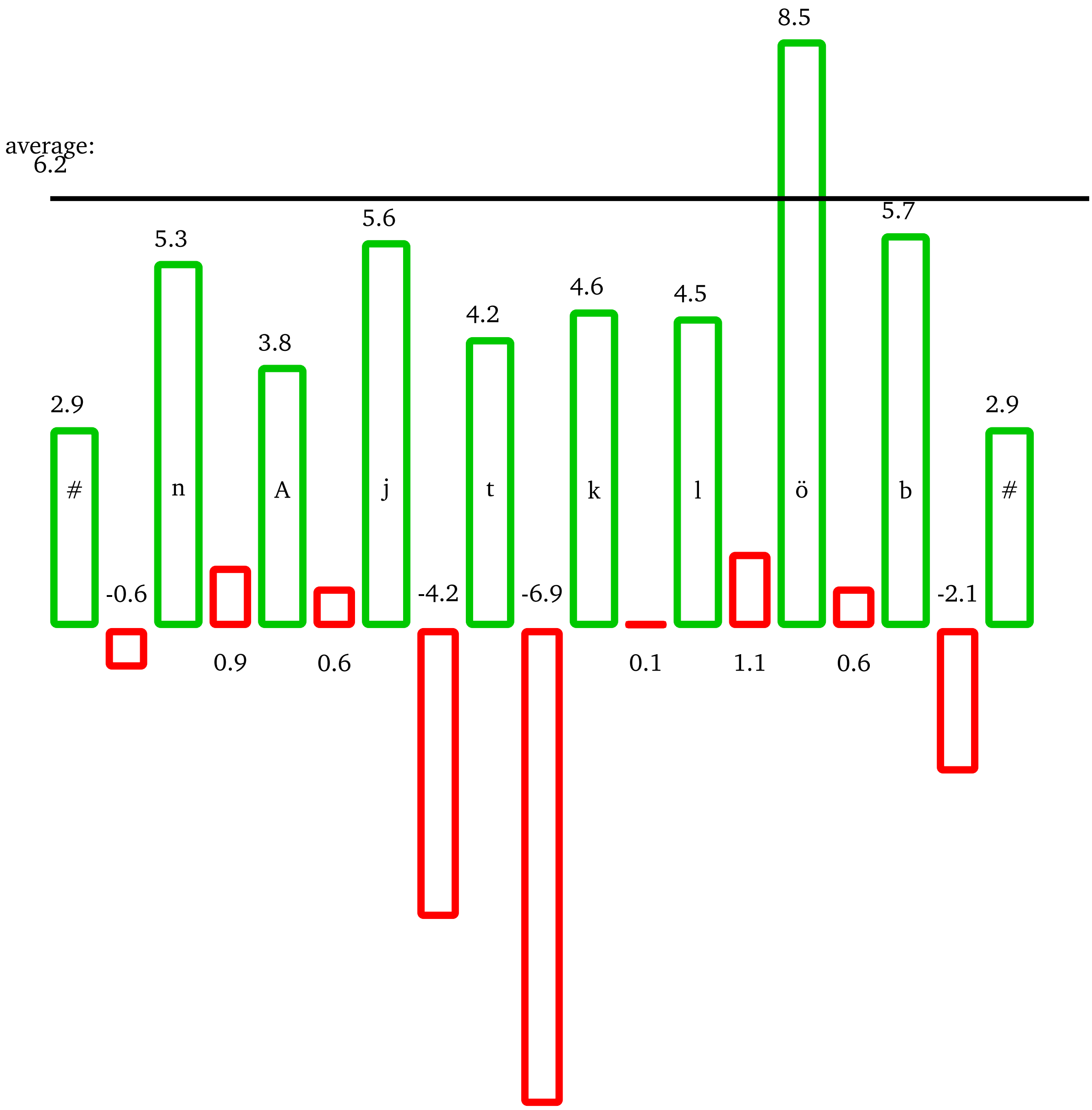
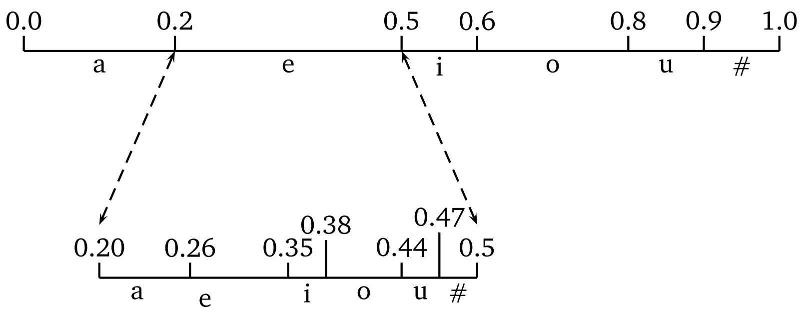
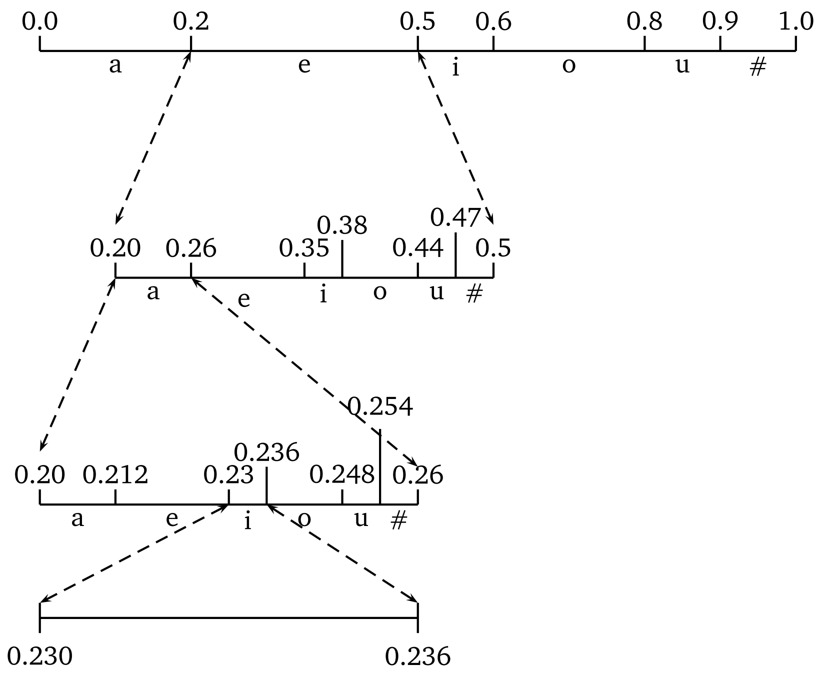


Fig. 3.7: Nightclub, in French



Symbol	Probability	Range
a	0.2	[0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
$	0.1	[0.9, 1.0)



3.2 Linear structure: bigram model and conditional probability

Unigram models describe the basic frequency of phonemes. Much of the phonological structure of languages, however, involves conditions on sequences of phonemes, which goes beyond the descriptive purview of unigram models. The natural way to encode this information is to use a bigram model, which is to say, to use as the probability for a given phoneme its probability in given context.[16]

One of the simplest models along these lines conditions the probability of a phoneme on its left-hand neighbor in the word. Because the initial segment of a word, $w[1]$, does not have a left-neighbor, it is conventional to define $w[0]$ as the boundary symbol #. Informally speaking,

[16]There has been an unfortunate inconsistency in the use of the terms 0-order and 1st-order Markov models over the years. The older tradition of usage defines a 0-order Markov model as one assigning a uniform distribution over symbols, and a 1st-order Markov model as one in which each symbol is assigned a probability independent of context—what we call here a *unigram model*. The newer tradition of usage, which we follow here, uses the term 0-order Markov model the unigram model and the term 1st-order model for models with one symbol of context (e.g., bigram models).

the *conditional probability* of phoneme b immediately following a, where a is the left-neighbor or # if b is word-initial, is calculated as in (3.5):

$$p(b \mid a) = \frac{Count(ab)}{Count(a)} \tag{3.5}$$

where $Count(ab)$ denotes the number of times that b occurs in context a in the word-set and $Count(a)$ denotes the number of times that context a occurs.[17]

3.2.1 Simple example

Imagine a language for which we have just 3 words:

#bada#
#banda#
#nand#

16 letters: we do not count the first #; its probability of being *there* is 1.0. We do count the final #, because its occurrence where it is is not predictable, and its presence allows a generalization to be made about what letters are likely to occur word-finally.

Here and throughout, rows indicate the first (preceding) letter.

count

first	second letter					
letter	a	b	d	n	#	count
a	0	0	1	2	2	5
b	2	0	0	0	0	2
d	2	0	0	0	1	3
n	1	0	2	0	0	3
#	0	2	0	1	0	3

		second letter				
		a	b	d	n	#
	a	0	0	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{2}{16}$
frequency	b	$\frac{2}{16}$	0	0	0	0
	d	$\frac{2}{16}$	0	0	0	$\frac{1}{16}$
	n	$\frac{1}{16}$	0	$\frac{2}{16}$	0	0
	#	0 ‘	$\frac{2}{16}$	0	$\frac{1}{16}$	0

[17] The right way to say this is:

$$p(w[i]{=}b \mid w[i-1]{=}a) = \frac{p(w[i-1] = a \,\&\, w[i] = b)}{p(w[i-1] = a)}. \tag{3.6}$$

probability of second letter, given the first (preceding): (rows sum to 1.0)

	second letter				
	a	b	d	n	#
a	0	0	$\frac{1}{5}$	$\frac{2}{5}$	$\frac{2}{5}$
b	1	0	0	0	0
d	$\frac{2}{3}$	0	0	0	$\frac{1}{3}$
n	$\frac{1}{3}$	0	$\frac{2}{3}$	0	0
#	0	$\frac{2}{3}$	0	$\frac{1}{3}$	0

probability of first letter, given the second (following): (columns sum to 1.0)

	second letter				
	a	b	d	n	#
a	0	0	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$
b	$\frac{2}{5}$	0	0	0	0
d	$\frac{2}{5}$	0	0	0	$\frac{1}{3}$
n	$\frac{1}{5}$	0	$\frac{2}{3}$	0	0
#	0	1	0	$\frac{1}{3}$	0

3.3 Logarithms

A considerable advantage comes now from using logarithms: it allows us to easily express what the advantage is of the bigram model over the unigram model. The change in the log probability computed under the unigram and the bigram models is precisely equal to another quantity of particular interest, the *mutual information*, defined as in (3.7).

$$\begin{aligned} MI(a;b) = \log \frac{p(ab)}{p(a)p(b)} &= \log\, p(ab) - \log\, p(a) - \log\, p(b) \\ &= -plog(ab) + plog(a) + plog(b) \end{aligned} \tag{3.7}$$

If $p(ab) = \frac{Count(ab)}{|S|}$ is the probability of the pair ab and $p(a)p(b)$ is the product of the symbol's individual probabilities, then the mutual information between a and b is the log of the ratio of these quantities. The probability of a joint event, such as the sequence ab, is equal to the product of the individual probabilities just in case the two events are independent of each other (this being the definition of independence), so the ratio here takes the value 1 just in case the two events are independent.

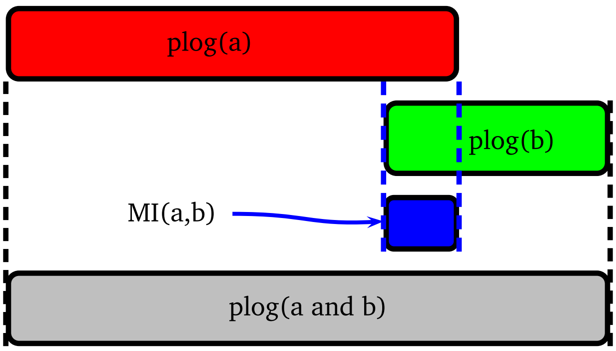
[18] If the probability sequence of the phonemes is greater than the product of the individual probabilities, then the structure involved in the model being explored pulls the two events together,

[18] mutual information, pointwise mutual information, weighted mutual information

while if the probability of the phonemes together is less than the product, the structure at hand is responsible for them repelling each other, so to speak. By taking the logarithm of this ratio, we translate attraction to a positive value, repelling to a negative value, and independence to a zero value. (When we are calculating this quantity for particular symbols, the term *pointwise mutual information* is often used, and then the term *mutual information* is used to describe the average pointwise mutual information as we average over all pairs of elements, each pair weighted by its probability. The weighted mutual information of a pair is the pair's MI times its count.)

Just as important, the mutual information is exactly the difference between the unigram and bigram model's log probability. This is shown in (3.8).

$$\begin{aligned}\sum_{i=1}^{|w|} \log p(w[i] \mid w[i-1]) &= \sum_{i=1}^{|w|} \log \frac{p(w[i]\, w[i-1])}{p(w[i-1])} \\ &= \sum_{i=1}^{|w|} \log p(w[i]) + \sum_{i=1}^{|w|} \log \frac{p(w[i]\, w[i-1])}{p(w[i-1])p(w[i])} \\ &= \sum_{i=1}^{|w|} \log p(w[i]) + MI(w[i-1];\, w[i])\end{aligned} \tag{3.8}$$



For a concrete illustration we return to our English word list from Table 3.2. Our English data set contains 54 phonemes, and thus there are $54^2 = 2,916$ possible bigrams. Consider, in Table 3.3, the way that the bigram model enriches the evaluation of the English data by taking 2-word slices at six points along the ranking of all 63,000 words according to their average bigram plog.

With the bigram model, we obtain a set of parameters that describe the phonological well-formedness (in terms of 'typicality') to a second order degree of detail. If there are P phonemes in the language, then the number of parameters for the unigram and bigram models together is $P + P^2$. Each setting of values (weights) for the parameters assigns a probability to a corpus, and

rank	orthography	phonemes	*avg. plog*$_2$
1	the	ðə	1.93
2	hand	hǽnd	2.15
12,640			
12,640	plumbing	plʌ́mĭŋ	3.71
12,642	Friday	fraýdĭ	3.71
25,281	tolls	tólz	4.01
25,282	recorder	r ĭ k ó r d ɚ̆	4.01
37,922	overburdened	óvɚ̆bɚ́dənd	4.32
37,923	Australians	ɔ̆streýlyənz	4.32
50,563	retire	rĭtaýr	4.75
50,564	poorer	púrɚ̆	4.75
63,200	eh	ɛ́	9.07
63,201	Oahu	óáhŭ	9.21

Tab. 3.3: English words ranked by average plog in the bigram model

the degree of success acheived by a set of parameters with weightings can be measured by that probability: the higher the probability, the more successful the characterization.[19]

3.3.0.1 Some notation we're using

$$N = \sum_{l \in a..z} [l]$$

$$pr(S[i] = w_j) = \frac{[w_j]}{N}$$

$$pr(S[i] = h | S[i-1] = t)$$

or (sorry, this is really a terrible abuse of notation)

$$pr_2(h|t) = \frac{[th]}{[t]}$$

[19] One striking characteristic of probabilistic phonology of the 1950s (e.g., Cherry, Halle, and Jakobson (1953) [?]; [?]; etc.), compared with what we attempt to do here (or Coleman and Pierrehumbert (1977) ([?])), is the focus in that early work on average values over an entire corpus. The clearest example of this is the emphasis on calculating the entropy of a language under various models. The entropy is the weighted average of the inverse log frequency, and each word in the lexicon contributes to its computation in proportion to the word's frequency in the language. By contrast, we are not only interested in these "ensemble averages," we are also interested in how some words (or subgroups of words) differ from other words.

Remember that plog (x) equals $-log_2 pr(x)$ if we are talking about a particular distribution pr over a set containing x, and that is clear from context; else plog(x) = $-log_2(x)$ and $0 < x \leq 1$.

$$plog(h|t) = plog(\frac{[th]}{[t]}) = plog(\frac{fr(th)}{fr(t)}$$

because we divide both numerator and denominator by N.

Remember comparing observed to expected? Here, *expected* is typically taken to be *if there were no structure, and distributions were independent.*

Pointwise mutual information of the ordered pair ab is $log \frac{p(ab)}{p(a)p(b)} = log \frac{p(ab)}{p(a)} - \ logp(b)$, or

$$log \frac{p(ab)}{p(a)p(b)} = -plog \frac{p(ab)}{p(a)} + plogp_1(b)$$

That tells us that *the bigram plog of* b *is equal to the unigram plog of* b *minus the PMI of (ab)*:

$$plog_2(b|a) = plog_1(b) - log \frac{p(ab)}{p(a)p(b)}$$

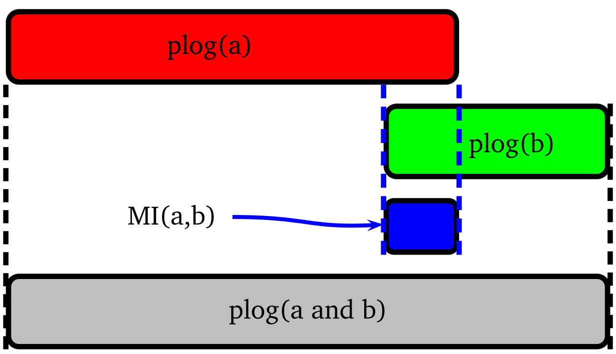
	word	count	frequency		word	count	frequency
1	the	69903	0.068271	11	for	9472	0.009251
2	of	36341	0.035493	12	it	9082	0.008870
3	and	28772	0.028100	13	with	7277	0.007107
4	to	26113	0.025503	14	as	7244	0.007075
5	a	23309	0.022765	15	his	6992	0.006829
6	in	21304	0.020807	16	on	6732	0.006575
7	that	10780	0.010528	17	be	6368	0.006219
8	is	10100	0.009864	18	s	5958	0.005819
9	was	9814	0.009585	19	I	5909	0.005771
10	he	9799	0.009570	20	at	5368	0.005243

Tab. 3.4: Top of the unigram distribution for the Brown Corpus.

	word	count	count / 69,936		word	count	count / 69,936
0	first	664	0.00949				
1	same	629	0.00899	11	way	239	0.00342
2	other	419	0.00599	12	old	234	0.00335
3	most	419	0.00599	13	last	223	0.00319
4	new	398	0.00569	14	house	216	0.00309
5	world	393	0.00562	15	man	214	0.00306
6	united	385	0.00551	16	next	210	0.00300
7	state	271	0.00418	17	end	206	0.00295
8	two	267	0.00382	18	fact	194	0.00277
9	only	260	0.00372	19	whole	190	0.00272
10	time	250	0.00357	20	American	184	0.00263

Tab. 3.5: Top of the Brown Corpus for words following *the*.

3.3.0.2 Words



	word	count	count / 36,388		word	count	count / 36,388
1	the	9724	0.267	11	her	252	0.00693
2	a	1473	0.0405	12	our	251	0.00690
3	his	810	0.0223	13	its	229	0.00629
4	this	553	0.01520	14	it	205	0.00563
5	their	342	0.00940	15	that	156	0.00429
6	course	324	0.00890	16	such	140	0.00385
7	these	306	0.00841	17	those	135	0.00371
8	them	292	0.00802	18	my	128	0.00352
9	an	276	0.00758	19	which	124	0.00341
10	all	256	0.00704	20	new	118	0.00324

Tab. 3.6: Top of the Brown Corpus for words following *of*.

	word	count	count / 69,936		word	count	count / 69,936
1	of	9724	0.139	11	from	1415	0.0202
2	.	6201	0.0887	12	that	1397	0.0199
3	in	6027	0.0862	13	by	1349	0.0193
4	,	3836	0.0548	14	is	799	0.0114
5	to	3485	0.0498	15	as	766	0.0109
6	on	2469	0.0353	16	into	675	0.00965
7	and	2254	0.0322	17	was	533	0.00762
8	for	1850	0.0264	18	all	430	0.00615
9	at	1657	0.0237	19	when	418	0.00597
10	with	1536	0.0219	20	but	389	0.00556

Tab. 3.7: Top of the Brown Corpus for words preceding *the*.

	word	count	count / 69,936		word	count	count / 69,936
1	of	10861	0.155	11	for	598	0.00855
2	.	4578	0.0655	12	were	386	0.00552
3	,	4437	0.0634	13	with	370	0.00529
4	and	2473	0.0354	14	on	368	0.00526
5	to	1188	0.0170	15	states	366	0.00523
6	’	1106	0.0158	16	had	340	0.00486
7	in	1082	0.0155	17	are	330	0.00472
8	is	1049	0.0150	18	as	299	0.00428
9	was	950	0.0136	19	at	287	0.00410
10	that	888	0.0127	20	or	284	0.00406

Tab. 3.8: Top of the Brown Corpus for words 2 to the right of *the*.

3.4 All words

Words, sorted by frequency

rank	word	count	frequency	plog
1	the	179173	0.070	3.846
2	,	170882	0.066	3.915
3	.	110224	0.043	4.547
4	of	106057	0.041	4.603
5	and	79108	0.031	5.026
6	in	68995	0.027	5.223
7	a	42377	0.016	5.926
8	to	39522	0.015	6.027
9	)	30051	0.012	6.422
10	(	30029	0.012	6.423
11	is	22204	0.009	6.859
12	by	19874	0.008	7.019
13	was	18721	0.007	7.105
14	as	18073	0.007	7.156
15	for	15699	0.006	7.359
16	are	14412	0.006	7.482
17	;	13294	0.005	7.599
18	on	12487	0.005	7.689
19	with	12481	0.005	7.690
20	that	12153	0.005	7.728
21	or	11468	0.004	7.812
22	from	10973	0.004	7.876
23	he	10751	0.004	7.905
24	his	10116	0.004	7.993
25	an	8432	0.003	8.256

Word pairs, sorted by bigram frequency

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.629
2	, and	22761	0.008855	6.819	2.121	48283.513
3	. the	19962	0.007766	7.009	1.385	27649.829
4	in the	17633	0.006860	7.188	1.882	33185.650
5	, the	11492	0.004471	7.805	-0.044	-506.398
6	) ,	11492	0.004471	7.805	2.532	29095.481
7	. in	8910	0.003466	8.172	1.598	14239.550
8	and the	7801	0.003035	8.364	0.508	3964.006
9	to the	7178	0.002792	8.484	1.389	9971.902
10	by the	6424	0.002499	8.644	2.221	14267.029
11	) .	5479	0.002132	8.874	2.096	11482.449
12	on the	5117	0.001991	8.973	2.563	13115.775
13	, which	4465	0.001737	9.169	3.094	13815.228
14	, in	4334	0.001686	9.212	-0.074	-321.299
15	, a	3992	0.001553	9.331	0.510	2037.870
16	of a	3654	0.001422	9.458	1.071	3913.485
17	, or	3427	0.001333	9.551	2.176	7457.119
18	. he	3400	0.001323	9.562	2.890	9826.966
19	as a	3333	0.001297	9.591	3.491	11636.471
20	from the	3330	0.001295	9.592	2.130	7092.465
21	with the	3265	0.001270	9.621	1.916	6254.613
22	for the	3094	0.001204	9.698	1.507	4662.985
23	as the	2993	0.001164	9.746	1.256	3759.392
24	, but	2909	0.001132	9.787	3.331	9689.344
25	at the	2875	0.001118	9.804	2.404	6911.103

Word pairs, sorted by repelling bigram mutual information

rank	bigram	count	frequency	plog	MI	weighted MI
491124	the the	1	0.000000	21.294	-13.601	-13.6
724434	and .	1	0.000000	21.294	-11.720	-11.7
241133	in of	1	0.000000	21.294	-11.467	-11.5
85092	the .	4	0.000002	19.294	-10.900	-43.6
469339	in in	1	0.000000	21.294	-10.847	-10.8
90186	, .	4	0.000002	19.294	-10.831	-43.3
472961	(.	1	0.000000	21.294	-10.323	-10.3
199147	the)	2	0.000001	20.294	-10.025	-20.0
93866	of in	3	0.000001	19.709	-9.882	-29.6
675385	and)	1	0.000000	21.294	-9.845	-9.8
22555	the ,	13	0.000005	17.593	-9.832	-127.8
744829	, ;	1	0.000000	21.294	-9.780	-9.8
278561	the or	1	0.000000	21.294	-9.635	-9.6
643287	of for	1	0.000000	21.294	-9.332	-9.3
118452	a of	3	0.000001	19.709	-9.179	-27.5
395579	; .	1	0.000000	21.294	-9.147	-9.1
93085	to .	3	0.000001	19.709	-9.134	-27.4
399701	as and	1	0.000000	21.294	-9.112	-9.1
343518	at ,	1	0.000000	21.294	-9.017	-9.0
90508	,)	4	0.000002	19.294	-8.957	-35.8
45181	the a	7	0.000003	18.486	-8.713	-61.0
84208	of to	4	0.000002	19.294	-8.664	-34.7
364211	in ;	1	0.000000	21.294	-8.471	-8.5
287576	or and	1	0.000000	21.294	-8.456	-8.5
612868	))	1	0.000000	21.294	-8.449	-8.4

Word pairs, sorted by attracting bigram mutual information

rank	bigram	count	frequency	plog	MI	weighted MI
146446	capo d'istria	2	0.000001	20.294	22.301	44.6
147015	guillaine barré	2	0.000001	20.294	22.301	44.6
157955	angina pectoris	2	0.000001	20.294	22.301	44.6
164650	governador valadares	2	0.000001	20.294	22.301	44.6
219935	dosso dossi	2	0.000001	20.294	22.301	44.6
223527	akutagawa ryûnosuke	2	0.000001	20.294	22.301	44.6
145303	chikamatsu monzaemon	2	0.000001	20.294	21.301	42.6
149146	dandie dinmont	2	0.000001	20.294	21.301	42.6
205812	fukuzawa yukichi	2	0.000001	20.294	21.301	42.6
225440	petrus christus	1	0.000000	21.294	21.301	21.3
225815	cactus-thorn "tool	1	0.000000	21.294	21.301	21.3
225993	befuddled underachiever	1	0.000000	21.294	21.301	21.3
226959	hesperiphona vespertina	1	0.000000	21.294	21.301	21.3
227239	uuno kailas	1	0.000000	21.294	21.301	21.3
227246	kostes palamas	1	0.000000	21.294	21.301	21.3
227365	gian-carlo menotti	1	0.000000	21.294	21.301	21.3
227391	siegbert tarrasch	1	0.000000	21.294	21.301	21.3
227663	seraphima astafieva	1	0.000000	21.294	21.301	21.3
228582	pogonias cromis	1	0.000000	21.294	21.301	21.3
228790	tetramorium caespitum	1	0.000000	21.294	21.301	21.3
230101	tursiops truncatus	1	0.000000	21.294	21.301	21.3
230889	clapham sect-a	1	0.000000	21.294	21.301	21.3
231365	tetraborate decahydrate-a	1	0.000000	21.294	21.301	21.3
231716	"carrying amount"	1	0.000000	21.294	21.301	21.3
231936	toivo pekkanen	1	0.000000	21.294	21.301	21.3

Word pairs, sorted by attracting bigram weighted mutual information

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	1	0.000000	6.391	2.059	63066.6
2	, and	1	0.000000	6.819	2.121	48283.5
4	in the	1	0.000000	7.188	1.882	33185.7
6	) ,	1	0.000000	7.805	2.532	29095.5
3	. the	1	0.000000	7.009	1.385	27649.8
26	u .s	1	0.000000	9.841	9.825	27540.6
30	such as	1	0.000000	10.049	6.278	15237.3
10	by the	1	0.000000	8.644	2.221	14267.0
7	. in	1	0.000000	8.172	1.598	14239.6
13	, which	1	0.000000	9.169	3.094	13815.2
12	on the	1	0.000000	8.973	2.563	13115.8
29	he was	1	0.000000	10.009	5.001	12478.7
72	united states	1	0.000000	11.028	9.949	12247.0
27	.s .	1	0.000000	9.899	4.486	12081.3
19	as a	1	0.000000	9.591	3.491	11636.5
11	) .	1	0.000000	8.874	2.096	11482.4
34	it is	1	0.000000	10.181	4.937	10934.4
68	more than	1	0.000000	10.982	8.215	10440.7
9	to the	1	0.000000	8.484	1.389	9971.9
18	. he	1	0.000000	9.562	2.890	9827.0
24	, but	1	0.000000	9.787	3.331	9689.3
77	have been	1	0.000000	11.042	7.724	9415.8
99	new york	1	0.000000	11.345	9.381	9268.1
48	(see	1	0.000000	10.581	5.477	9190.4
63	known as	1	0.000000	10.842	6.339	8874.4

Word pairs, sorted by bigram count

3.5 the

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
3	. the	19962	0.007766	7.009	1.385	27649.8
4	in the	17633	0.006860	7.188	1.882	33185.7
5	, the	11492	0.004471	7.805	-0.044	-506.4
8	and the	7801	0.003035	8.364	0.508	3964.0
9	to the	7178	0.002792	8.484	1.389	9971.9
10	by the	6424	0.002499	8.644	2.221	14267.0
12	on the	5117	0.001991	8.973	2.563	13115.8
20	from the	3330	0.001295	9.592	2.130	7092.5
21	with the	3265	0.001270	9.621	1.916	6254.6
22	for the	3094	0.001204	9.698	1.507	4663.0
23	as the	2993	0.001164	9.746	1.256	3759.4
25	at the	2875	0.001118	9.804	2.404	6911.1
32	is the	2367	0.000921	10.085	0.621	1468.8
35	the first	2197	0.000855	10.192	3.097	6803.4
40	the u	2026	0.000788	10.309	3.366	6819.6
41	during the	1976	0.000769	10.345	3.157	6238.2
46	; the	1777	0.000691	10.498	0.947	1682.8
51	the most	1605	0.000624	10.645	2.485	3988.3
60	was the	1431	0.000557	10.811	0.141	201.3
67	the city	1280	0.000498	10.972	2.469	3159.9
69	the united	1250	0.000486	11.006	3.546	4433.0
73	that the	1231	0.000479	11.028	0.547	673.1
78	the american	1218	0.000474	11.043	2.495	3039.0
88	the french	1126	0.000438	11.157	2.830	3186.4

Word pairs, with *the* sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
4	in the	17633	0.006860	7.188	1.882	33185.7
3	. the	19962	0.007766	7.009	1.385	27649.8
10	by the	6424	0.002499	8.644	2.221	14267.0
12	on the	5117	0.001991	8.973	2.563	13115.8
9	to the	7178	0.002792	8.484	1.389	9971.9
20	from the	3330	0.001295	9.592	2.130	7092.5
25	at the	2875	0.001118	9.804	2.404	6911.1
40	the u	2026	0.000788	10.309	3.366	6819.6
35	the first	2197	0.000855	10.192	3.097	6803.4
21	with the	3265	0.001270	9.621	1.916	6254.6
41	during the	1976	0.000769	10.345	3.157	6238.2
22	for the	3094	0.001204	9.698	1.507	4663.0
69	the united	1250	0.000486	11.006	3.546	4433.0
51	the most	1605	0.000624	10.645	2.485	3988.3
8	and the	7801	0.003035	8.364	0.508	3964.0
95	the same	1010	0.000393	11.313	3.791	3829.4
23	as the	2993	0.001164	9.746	1.256	3759.4
88	the french	1126	0.000438	11.157	2.830	3186.4
67	the city	1280	0.000498	10.972	2.469	3159.9
78	the american	1218	0.000474	11.043	2.495	3039.0
118	the late	868	0.000338	11.532	3.483	3022.8
109	the british	923	0.000359	11.443	3.050	2815.4
103	among the	938	0.000365	11.420	2.710	2542.3
139	the university	754	0.000293	11.735	2.810	2118.5

Word pairs, with *the* on left side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
35	the first	2197	0.000855	10.192	3.097	6803.4
40	the u	2026	0.000788	10.309	3.366	6819.6
51	the most	1605	0.000624	10.645	2.485	3988.3
67	the city	1280	0.000498	10.972	2.469	3159.9
69	the united	1250	0.000486	11.006	3.546	4433.0
78	the american	1218	0.000474	11.043	2.495	3039.0
88	the french	1126	0.000438	11.157	2.830	3186.4
95	the same	1010	0.000393	11.313	3.791	3829.4
109	the british	923	0.000359	11.443	3.050	2815.4
118	the late	868	0.000338	11.532	3.483	3022.8
131	the new	803	0.000312	11.644	1.660	1332.9
134	the world	794	0.000309	11.661	2.245	1782.4
135	the early	782	0.000304	11.683	2.483	1941.9
139	the university	754	0.000293	11.735	2.810	2118.5
157	the english	672	0.000261	11.901	2.478	1665.0
163	the north	657	0.000256	11.934	2.497	1640.8
168	the great	648	0.000252	11.954	2.044	1324.6
174	the state	633	0.000246	11.988	2.045	1294.2
178	the country	624	0.000243	12.008	3.164	1974.4
181	the national	610	0.000237	12.041	2.346	1431.3
184	the roman	598	0.000233	12.070	2.420	1446.9
188	the other	579	0.000225	12.116	0.787	455.8
211	the german	527	0.000205	12.252	2.579	1359.1
214	the south	522	0.000203	12.266	2.329	1215.9
216	the 19th	521	0.000203	12.268	3.358	1749.4

Word pairs, with *the* on left side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
40	the u	2026	0.000788	10.309	3.366	6819.6
35	the first	2197	0.000855	10.192	3.097	6803.4
69	the united	1250	0.000486	11.006	3.546	4433.0
51	the most	1605	0.000624	10.645	2.485	3988.3
95	the same	1010	0.000393	11.313	3.791	3829.4
88	the french	1126	0.000438	11.157	2.830	3186.4
67	the city	1280	0.000498	10.972	2.469	3159.9
78	the american	1218	0.000474	11.043	2.495	3039.0
118	the late	868	0.000338	11.532	3.483	3022.8
109	the british	923	0.000359	11.443	3.050	2815.4
139	the university	754	0.000293	11.735	2.810	2118.5
178	the country	624	0.000243	12.008	3.164	1974.4
135	the early	782	0.000304	11.683	2.483	1941.9
134	the world	794	0.000309	11.661	2.245	1782.4
216	the 19th	521	0.000203	12.268	3.358	1749.4
157	the english	672	0.000261	11.901	2.478	1665.0
163	the north	657	0.000256	11.934	2.497	1640.8
239	the middle	483	0.000188	12.378	3.256	1572.8
237	the principal	483	0.000188	12.378	3.202	1546.5
232	the term	493	0.000192	12.348	3.075	1515.8
184	the roman	598	0.000233	12.070	2.420	1446.9
246	the end	471	0.000183	12.414	3.059	1440.9
181	the national	610	0.000237	12.041	2.346	1431.3
211	the german	527	0.000205	12.252	2.579	1359.1
256	the second	455	0.000177	12.464	2.938	1337.0

Word pairs, with *the* on right side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
3	. the	19962	0.007766	7.009	1.385	27649.8
4	in the	17633	0.006860	7.188	1.882	33185.7
5	, the	11492	0.004471	7.805	-0.044	-506.4
8	and the	7801	0.003035	8.364	0.508	3964.0
9	to the	7178	0.002792	8.484	1.389	9971.9
10	by the	6424	0.002499	8.644	2.221	14267.0
12	on the	5117	0.001991	8.973	2.563	13115.8
20	from the	3330	0.001295	9.592	2.130	7092.5
21	with the	3265	0.001270	9.621	1.916	6254.6
22	for the	3094	0.001204	9.698	1.507	4663.0
23	as the	2993	0.001164	9.746	1.256	3759.4
25	at the	2875	0.001118	9.804	2.404	6911.1
32	is the	2367	0.000921	10.085	0.621	1468.8
41	during the	1976	0.000769	10.345	3.157	6238.2
46	; the	1777	0.000691	10.498	0.947	1682.8
60	was the	1431	0.000557	10.811	0.141	201.3
73	that the	1231	0.000479	11.028	0.547	673.1
89	are the	1087	0.000423	11.207	0.121	131.9
101	after the	956	0.000372	11.393	2.119	2025.3
102	into the	951	0.000370	11.400	1.987	1889.6
103	among the	938	0.000365	11.420	2.710	2542.3
132	between the	803	0.000312	11.644	2.272	1824.8
133	when the	795	0.000309	11.659	1.853	1472.7
138	under the	761	0.000296	11.722	2.457	1869.7

Word pairs, with *the* on right side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
4	in the	17633	0.006860	7.188	1.882	33185.7
3	. the	19962	0.007766	7.009	1.385	27649.8
10	by the	6424	0.002499	8.644	2.221	14267.0
12	on the	5117	0.001991	8.973	2.563	13115.8
9	to the	7178	0.002792	8.484	1.389	9971.9
20	from the	3330	0.001295	9.592	2.130	7092.5
25	at the	2875	0.001118	9.804	2.404	6911.1
21	with the	3265	0.001270	9.621	1.916	6254.6
41	during the	1976	0.000769	10.345	3.157	6238.2
22	for the	3094	0.001204	9.698	1.507	4663.0
8	and the	7801	0.003035	8.364	0.508	3964.0
23	as the	2993	0.001164	9.746	1.256	3759.4
103	among the	938	0.000365	11.420	2.710	2542.3
101	after the	956	0.000372	11.393	2.119	2025.3
102	into the	951	0.000370	11.400	1.987	1889.6
138	under the	761	0.000296	11.722	2.457	1869.7
132	between the	803	0.000312	11.644	2.272	1824.8
148	through the	710	0.000276	11.822	2.455	1743.3
46	; the	1777	0.000691	10.498	0.947	1682.8
133	when the	795	0.000309	11.659	1.853	1472.7
32	is the	2367	0.000921	10.085	0.621	1468.8
200	over the	550	0.000214	12.190	2.485	1366.7
196	against the	561	0.000218	12.162	2.430	1363.4
304	throughout the	402	0.000156	12.643	3.153	1267.3

3.6 of

Word pairs, sorted by bigram count

with *of*

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
16	of a	3654	0.001422	9.458	1.071	3913.5
47	one of	1707	0.000664	10.556	3.417	5832.7
54	of his	1550	0.000603	10.696	1.900	2945.7
65	part of	1296	0.000504	10.954	4.212	5458.6
81	number of	1178	0.000458	11.091	4.323	5092.5
122	, of	849	0.000330	11.564	-3.046	-2586.3
127	of which	816	0.000317	11.621	1.330	1085.5
136	of an	772	0.000300	11.701	1.158	893.6
137	because of	768	0.000299	11.709	3.326	2554.3
140	use of	753	0.000293	11.737	3.598	2709.4
143	university of	741	0.000288	11.760	3.541	2624.0
142	most of	741	0.000288	11.760	2.126	1575.7
151	of these	699	0.000272	11.844	2.604	1820.0
154	of their	695	0.000270	11.853	1.867	1297.8
159	of its	669	0.000260	11.908	1.773	1186.4
158	) of	669	0.000260	11.908	-0.883	-590.4
171	king of	646	0.000251	11.958	3.212	2074.7
173	of this	640	0.000249	11.972	1.802	1153.1
183	form of	603	0.000235	12.058	3.100	1869.6
198	development of	551	0.000214	12.188	3.657	2015.2
202	end of	547	0.000213	12.198	4.031	2205.2
215	of all	521	0.000203	12.268	2.049	1067.4
217	that of	521	0.000203	12.268	0.063	32.8
225	of about	504	0.000196	12.316	0.993	500.5

Word pairs, with *of* sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
47	one of	1707	0.000664	10.556	3.417	5832.7
65	part of	1296	0.000504	10.954	4.212	5458.6
81	number of	1178	0.000458	11.091	4.323	5092.5
16	of a	3654	0.001422	9.458	1.071	3913.5
54	of his	1550	0.000603	10.696	1.900	2945.7
140	use of	753	0.000293	11.737	3.598	2709.4
143	university of	741	0.000288	11.760	3.541	2624.0
137	because of	768	0.000299	11.709	3.326	2554.3
202	end of	547	0.000213	12.198	4.031	2205.2
171	king of	646	0.000251	11.958	3.212	2074.7
198	development of	551	0.000214	12.188	3.657	2015.2
183	form of	603	0.000235	12.058	3.100	1869.6
247	parts of	469	0.000182	12.420	3.982	1867.7
270	series of	441	0.000172	12.509	4.160	1834.6
151	of these	699	0.000272	11.844	2.604	1820.0
243	members of	477	0.000186	12.396	3.639	1735.7
315	variety of	390	0.000152	12.686	4.310	1681.0
142	most of	741	0.000288	11.760	2.126	1575.7
284	son of	428	0.000167	12.552	3.588	1535.8
375	consists of	348	0.000135	12.851	4.384	1525.8
362	member of	360	0.000140	12.802	4.124	1484.7
376	types of	348	0.000135	12.851	4.025	1400.5
325	capital of	384	0.000149	12.709	3.503	1345.3
384	site of	339	0.000132	12.888	3.940	1335.8

Word pairs, with *of* on left side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
16	of a	3654	0.001422	9.458	1.071	3913.5
54	of his	1550	0.000603	10.696	1.900	2945.7
127	of which	816	0.000317	11.621	1.330	1085.5
136	of an	772	0.000300	11.701	1.158	893.6
151	of these	699	0.000272	11.844	2.604	1820.0
154	of their	695	0.000270	11.853	1.867	1297.8
159	of its	669	0.000260	11.908	1.773	1186.4
173	of this	640	0.000249	11.972	1.802	1153.1
215	of all	521	0.000203	12.268	2.049	1067.4
225	of about	504	0.000196	12.316	0.993	500.5
381	of france	341	0.000133	12.880	2.544	867.4
405	of new	331	0.000129	12.923	1.138	376.6
407	of england	328	0.000128	12.936	2.444	801.5
438	of such	306	0.000119	13.036	0.738	225.8
487	of two	282	0.000110	13.154	1.271	358.3
502	of modern	273	0.000106	13.201	2.236	610.4
601	of many	237	0.000092	13.405	0.885	209.8
622	of great	231	0.000090	13.442	1.313	303.2
621	of other	231	0.000090	13.442	0.218	50.4
692	of life	210	0.000082	13.579	1.716	360.5
690	of king	210	0.000082	13.579	1.591	334.0
716	of human	205	0.000080	13.614	2.381	488.1
717	of one	205	0.000080	13.614	0.359	73.6
738	of several	199	0.000077	13.657	1.593	317.1

Word pairs, with *of* on left side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
1	of the	30635	0.011918	6.391	2.059	63066.6
16	of a	3654	0.001422	9.458	1.071	3913.5
54	of his	1550	0.000603	10.696	1.900	2945.7
151	of these	699	0.000272	11.844	2.604	1820.0
154	of their	695	0.000270	11.853	1.867	1297.8
159	of its	669	0.000260	11.908	1.773	1186.4
173	of this	640	0.000249	11.972	1.802	1153.1
127	of which	816	0.000317	11.621	1.330	1085.5
215	of all	521	0.000203	12.268	2.049	1067.4
136	of an	772	0.000300	11.701	1.158	893.6
381	of france	341	0.000133	12.880	2.544	867.4
407	of england	328	0.000128	12.936	2.444	801.5
502	of modern	273	0.000106	13.201	2.236	610.4
1102	of christ	147	0.000057	14.094	3.484	512.2
225	of about	504	0.000196	12.316	0.993	500.5
716	of human	205	0.000080	13.614	2.381	488.1
952	of god	167	0.000065	13.910	2.776	463.6
1015	of saint	157	0.000061	13.999	2.806	440.5
1214	of higher	137	0.000053	14.196	2.891	396.1
1432	of whom	122	0.000047	14.363	3.111	379.6
1666	of representatives	107	0.000042	14.552	3.522	376.8
405	of new	331	0.000129	12.923	1.138	376.6
787	of any	191	0.000074	13.716	1.935	369.6
692	of life	210	0.000082	13.579	1.716	360.5
487	of two	282	0.000110	13.154	1.271	358.3

Word pairs, with *of* on right side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
47	one of	1707	0.000664	10.556	3.417	5832.7
65	part of	1296	0.000504	10.954	4.212	5458.6
81	number of	1178	0.000458	11.091	4.323	5092.5
122	, of	849	0.000330	11.564	-3.046	-2586.3
137	because of	768	0.000299	11.709	3.326	2554.3
140	use of	753	0.000293	11.737	3.598	2709.4
143	university of	741	0.000288	11.760	3.541	2624.0
142	most of	741	0.000288	11.760	2.126	1575.7
158	) of	669	0.000260	11.908	-0.883	-590.4
171	king of	646	0.000251	11.958	3.212	2074.7
183	form of	603	0.000235	12.058	3.100	1869.6
198	development of	551	0.000214	12.188	3.657	2015.2
202	end of	547	0.000213	12.198	4.031	2205.2
217	that of	521	0.000203	12.268	0.063	32.8
243	members of	477	0.000186	12.396	3.639	1735.7
241	and of	477	0.000186	12.396	-2.767	-1319.8
247	parts of	469	0.000182	12.420	3.982	1867.7
266	some of	447	0.000174	12.489	1.836	820.6
270	series of	441	0.000172	12.509	4.160	1834.6
284	son of	428	0.000167	12.552	3.588	1535.8
305	center of	401	0.000156	12.646	2.968	1190.0
315	variety of	390	0.000152	12.686	4.310	1681.0
322	those of	385	0.000150	12.705	2.801	1078.4
321	many of	385	0.000150	12.705	1.585	610.4
325	capital of	384	0.000149	12.709	3.503	1345.3

Word pairs, with *of* on right side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
47	one of	1707	0.000664	10.556	3.417	5832.7
65	part of	1296	0.000504	10.954	4.212	5458.6
81	number of	1178	0.000458	11.091	4.323	5092.5
140	use of	753	0.000293	11.737	3.598	2709.4
143	university of	741	0.000288	11.760	3.541	2624.0
137	because of	768	0.000299	11.709	3.326	2554.3
202	end of	547	0.000213	12.198	4.031	2205.2
171	king of	646	0.000251	11.958	3.212	2074.7
198	development of	551	0.000214	12.188	3.657	2015.2
183	form of	603	0.000235	12.058	3.100	1869.6
247	parts of	469	0.000182	12.420	3.982	1867.7
270	series of	441	0.000172	12.509	4.160	1834.6
243	members of	477	0.000186	12.396	3.639	1735.7
315	variety of	390	0.000152	12.686	4.310	1681.0
142	most of	741	0.000288	11.760	2.126	1575.7
284	son of	428	0.000167	12.552	3.588	1535.8
375	consists of	348	0.000135	12.851	4.384	1525.8
362	member of	360	0.000140	12.802	4.124	1484.7
376	types of	348	0.000135	12.851	4.025	1400.5
325	capital of	384	0.000149	12.709	3.503	1345.3
384	site of	339	0.000132	12.888	3.940	1335.8
365	study of	359	0.000140	12.806	3.698	1327.6
393	means of	335	0.000130	12.906	3.918	1312.5
399	type of	334	0.000130	12.910	3.882	1296.6
472	seat of	289	0.000112	13.119	4.402	1272.3

3.7 to

with *to*

rank	bigram	count	frequency	plog	MI	weighted MI
9	to the	7178	0.002792	8.484	1.389	9971.9
56	to be	1491	0.000580	10.752	4.062	6057.0
75	to a	1225	0.000477	11.035	0.918	1125.1
116	, to	874	0.000340	11.522	-1.580	-1381.2
161	and to	663	0.000258	11.921	-0.868	-575.4
189	used to	577	0.000224	12.121	3.564	2056.4
193	according to	571	0.000222	12.136	6.023	3439.3
250	to have	466	0.000181	12.429	2.718	1266.4
290	began to	423	0.000165	12.569	4.626	1956.8
367	led to	358	0.000139	12.810	4.748	1699.8
371	. to	353	0.000137	12.830	-2.256	-796.3
396	returned to	334	0.000130	12.910	5.697	1902.8
401	applied to	333	0.000130	12.914	5.286	1760.3
463	to form	292	0.000114	13.104	3.478	1015.7
480	to his	286	0.000111	13.134	0.886	253.5
489	up to	280	0.000109	13.164	3.952	1106.5
512	to make	270	0.000105	13.217	4.923	1329.2
530	order to	262	0.000102	13.260	4.183	1096.0
568	) to	248	0.000096	13.339	-0.890	-220.7
581	to produce	242	0.000094	13.375	4.947	1197.2
604	is to	236	0.000092	13.411	-0.525	-123.9
619	able to	231	0.000090	13.442	5.811	1342.4
627	continued to	229	0.000089	13.454	4.715	1079.8
656	elected to	221	0.000086	13.506	4.513	997.3
676	to an	214	0.000083	13.552	0.731	156.4

Word pairs, with *to* sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
9	to the	7178	0.002792	8.484	1.389	9971.9
56	to be	1491	0.000580	10.752	4.062	6057.0
193	according to	571	0.000222	12.136	6.023	3439.3
189	used to	577	0.000224	12.121	3.564	2056.4
290	began to	423	0.000165	12.569	4.626	1956.8
396	returned to	334	0.000130	12.910	5.697	1902.8
401	applied to	333	0.000130	12.914	5.286	1760.3
367	led to	358	0.000139	12.810	4.748	1699.8
619	able to	231	0.000090	13.442	5.811	1342.4
512	to make	270	0.000105	13.217	4.923	1329.2
250	to have	466	0.000181	12.429	2.718	1266.4
581	to produce	242	0.000094	13.375	4.947	1197.2
75	to a	1225	0.000477	11.035	0.918	1125.1
489	up to	280	0.000109	13.164	3.952	1106.5
530	order to	262	0.000102	13.260	4.183	1096.0
627	continued to	229	0.000089	13.454	4.715	1079.8
463	to form	292	0.000114	13.104	3.478	1015.7
656	elected to	221	0.000086	13.506	4.513	997.3
950	said to	167	0.000065	13.910	5.714	954.3
809	related to	187	0.000073	13.747	4.997	934.4
762	addition to	195	0.000076	13.686	4.719	920.3
848	designed to	180	0.000070	13.802	4.893	880.8
999	attempt to	160	0.000062	13.972	5.495	879.2
931	went to	168	0.000065	13.901	5.161	867.1
1062	to prevent	152	0.000059	14.046	5.613	853.2

Word pairs, with *to* on left side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
9	to the	7178	0.002792	8.484	1.389	9971.9
56	to be	1491	0.000580	10.752	4.062	6057.0
75	to a	1225	0.000477	11.035	0.918	1125.1
250	to have	466	0.000181	12.429	2.718	1266.4
463	to form	292	0.000114	13.104	3.478	1015.7
480	to his	286	0.000111	13.134	0.886	253.5
512	to make	270	0.000105	13.217	4.923	1329.2
581	to produce	242	0.000094	13.375	4.947	1197.2
676	to an	214	0.000083	13.552	0.731	156.4
691	to their	210	0.000082	13.579	1.565	328.6
919	to provide	169	0.000066	13.893	4.828	816.0
968	to its	164	0.000064	13.936	1.169	191.8
965	to about	164	0.000064	13.936	0.797	130.8
1062	to prevent	152	0.000059	14.046	5.613	853.2
1064	to become	151	0.000059	14.055	3.866	583.8
1103	to that	147	0.000057	14.094	-0.338	-49.8
1124	to which	145	0.000056	14.114	0.262	38.0
1186	to establish	140	0.000054	14.164	5.652	791.3
1218	to those	137	0.000053	14.196	2.734	374.6
1437	to develop	121	0.000047	14.375	4.984	603.1
1485	to use	118	0.000046	14.411	2.348	277.1
1493	to all	118	0.000046	14.411	1.330	157.0
1503	to protect	117	0.000046	14.423	5.588	653.8
1549	to france	114	0.000044	14.461	2.387	272.1
1572	to other	112	0.000044	14.486	0.598	66.9

Word pairs, with *to* on left side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
9	to the	7178	0.002792	8.484	1.389	9971.9
56	to be	1491	0.000580	10.752	4.062	6057.0
512	to make	270	0.000105	13.217	4.923	1329.2
250	to have	466	0.000181	12.429	2.718	1266.4
581	to produce	242	0.000094	13.375	4.947	1197.2
75	to a	1225	0.000477	11.035	0.918	1125.1
463	to form	292	0.000114	13.104	3.478	1015.7
1062	to prevent	152	0.000059	14.046	5.613	853.2
919	to provide	169	0.000066	13.893	4.828	816.0
1186	to establish	140	0.000054	14.164	5.652	791.3
1503	to protect	117	0.000046	14.423	5.588	653.8
1437	to develop	121	0.000047	14.375	4.984	603.1
1717	to create	105	0.000041	14.579	5.668	595.2
1064	to become	151	0.000059	14.055	3.866	583.8
1820	to determine	99	0.000039	14.664	5.521	546.5
1628	to take	110	0.000043	14.512	4.657	512.3
2175	to obtain	87	0.000034	14.851	5.616	488.6
2148	to reduce	88	0.000034	14.834	5.291	465.6
2197	to maintain	87	0.000034	14.851	5.284	459.7
1944	to serve	95	0.000037	14.724	4.774	453.5
2544	to avoid	78	0.000030	15.008	5.793	451.8
1928	to give	95	0.000037	14.724	4.730	449.4
2407	to achieve	81	0.000032	14.954	5.488	444.5
2630	to ensure	76	0.000030	15.046	5.836	443.5
2542	to keep	78	0.000030	15.008	5.374	419.2

Word pairs, with *to* on right side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
116	, to	874	0.000340	11.522	-1.580	-1381.2
161	and to	663	0.000258	11.921	-0.868	-575.4
189	used to	577	0.000224	12.121	3.564	2056.4
193	according to	571	0.000222	12.136	6.023	3439.3
290	began to	423	0.000165	12.569	4.626	1956.8
367	led to	358	0.000139	12.810	4.748	1699.8
371	. to	353	0.000137	12.830	-2.256	-796.3
396	returned to	334	0.000130	12.910	5.697	1902.8
401	applied to	333	0.000130	12.914	5.286	1760.3
489	up to	280	0.000109	13.164	3.952	1106.5
530	order to	262	0.000102	13.260	4.183	1096.0
568	) to	248	0.000096	13.339	-0.890	-220.7
604	is to	236	0.000092	13.411	-0.525	-123.9
619	able to	231	0.000090	13.442	5.811	1342.4
627	continued to	229	0.000089	13.454	4.715	1079.8
656	elected to	221	0.000086	13.506	4.513	997.3
696	was to	209	0.000081	13.586	-0.454	-94.9
762	addition to	195	0.000076	13.686	4.719	920.3
786	similar to	191	0.000074	13.716	4.339	828.8
809	related to	187	0.000073	13.747	4.997	934.4
811	him to	186	0.000072	13.754	3.391	630.8
848	designed to	180	0.000070	13.802	4.893	880.8
855	them to	179	0.000070	13.810	3.070	549.5
931	went to	168	0.000065	13.901	5.161	867.1
950	said to	167	0.000065	13.910	5.714	954.3

Word pairs, with *to* on right side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
193	according to	571	0.000222	12.136	6.023	3439.3
189	used to	577	0.000224	12.121	3.564	2056.4
290	began to	423	0.000165	12.569	4.626	1956.8
396	returned to	334	0.000130	12.910	5.697	1902.8
401	applied to	333	0.000130	12.914	5.286	1760.3
367	led to	358	0.000139	12.810	4.748	1699.8
619	able to	231	0.000090	13.442	5.811	1342.4
489	up to	280	0.000109	13.164	3.952	1106.5
530	order to	262	0.000102	13.260	4.183	1096.0
627	continued to	229	0.000089	13.454	4.715	1079.8
656	elected to	221	0.000086	13.506	4.513	997.3
950	said to	167	0.000065	13.910	5.714	954.3
809	related to	187	0.000073	13.747	4.997	934.4
762	addition to	195	0.000076	13.686	4.719	920.3
848	designed to	180	0.000070	13.802	4.893	880.8
999	attempt to	160	0.000062	13.972	5.495	879.2
931	went to	168	0.000065	13.901	5.161	867.1
786	similar to	191	0.000074	13.716	4.339	828.8
1360	tend to	126	0.000049	14.316	6.008	757.0
1348	referred to	127	0.000049	14.305	5.953	756.1
1053	return to	153	0.000060	14.036	4.870	745.2
1018	subject to	157	0.000061	13.999	4.685	735.6
985	came to	162	0.000063	13.954	4.510	730.6
1306	due to	130	0.000051	14.271	5.522	717.8
1131	moved to	144	0.000056	14.124	4.948	712.5

3.8 house

with *house*

rank	bigram	count	frequency	plog	MI	weighted MI
499	house of	274	0.000107	13.196	3.515	963.1
637	the house	225	0.000088	13.480	2.474	556.7
3183	house ,	65	0.000025	15.271	0.751	48.8
5928	house (	40	0.000016	15.972	2.559	102.4
7924	a house	31	0.000012	16.339	1.695	52.5
8671	. house	29	0.000011	16.436	0.219	6.4
9295	house .	28	0.000011	16.486	0.169	4.7
11715	opera house	23	0.000009	16.770	9.080	208.8
12045	house in	22	0.000009	16.834	0.497	10.9
17176	house and	16	0.000006	17.294	-0.160	-2.6
19327	lower house	15	0.000006	17.387	7.167	107.5
19060	each house	15	0.000006	17.387	5.506	82.6
22523	white house	13	0.000005	17.593	6.510	84.6
34760	court house	9	0.000004	18.124	5.469	49.2
35512	and house	8	0.000003	18.294	-1.160	-9.3
43943	house was	7	0.000003	18.486	0.726	5.1
53432	royal house	6	0.000002	18.709	5.674	34.0
55523	house may	6	0.000002	18.709	3.154	18.9
51674	house on	6	0.000002	18.709	1.088	6.5
54336	house are	6	0.000002	18.709	0.881	5.3
64761	upper house	5	0.000002	18.972	6.389	31.9
62681	either house	5	0.000002	18.972	5.317	26.6
59445	state house	5	0.000002	18.972	3.322	16.6
63693	(house	5	0.000002	18.972	-0.441	-2.2
66422	house to	5	0.000002	18.972	-0.837	-4.2

Word pairs, with *house* sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
499	house of	274	0.000107	13.196	3.515	963.1
637	the house	225	0.000088	13.480	2.474	556.7
11715	opera house	23	0.000009	16.770	9.080	208.8
19327	lower house	15	0.000006	17.387	7.167	107.5
5928	house (	40	0.000016	15.972	2.559	102.4
22523	white house	13	0.000005	17.593	6.510	84.6
19060	each house	15	0.000006	17.387	5.506	82.6
7924	a house	31	0.000012	16.339	1.695	52.5
34760	court house	9	0.000004	18.124	5.469	49.2
3183	house ,	65	0.000025	15.271	0.751	48.8
84487	random house	4	0.000002	19.294	9.157	36.6
72133	hull house	4	0.000002	19.294	8.982	35.9
53432	royal house	6	0.000002	18.709	5.674	34.0
77031	house arrest	4	0.000002	19.294	8.497	34.0
64761	upper house	5	0.000002	18.972	6.389	31.9
73936	customs house	4	0.000002	19.294	6.952	27.8
62681	either house	5	0.000002	18.972	5.317	26.6
162587	carlton house	2	0.000001	20.294	11.526	23.1
109679	ruling house	3	0.000001	19.709	7.321	22.0
154063	somerset house	2	0.000001	20.294	9.652	19.3
55523	house may	6	0.000002	18.709	3.154	18.9
162648	manor house	2	0.000001	20.294	9.411	18.8
177819	house correspondent	2	0.000001	20.294	8.789	17.6
59445	state house	5	0.000002	18.972	3.322	16.6
81491	country house	4	0.000002	19.294	4.140	16.6

Word pairs, with *house* on left side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
499	house of	274	0.000107	13.196	3.515	963.1
3183	house ,	65	0.000025	15.271	0.751	48.8
5928	house (	40	0.000016	15.972	2.559	102.4
9295	house .	28	0.000011	16.486	0.169	4.7
12045	house in	22	0.000009	16.834	0.497	10.9
17176	house and	16	0.000006	17.294	-0.160	-2.6
43943	house was	7	0.000003	18.486	0.726	5.1
55523	house may	6	0.000002	18.709	3.154	18.9
51674	house on	6	0.000002	18.709	1.088	6.5
54336	house are	6	0.000002	18.709	0.881	5.3
66422	house to	5	0.000002	18.972	-0.837	-4.2
77031	house arrest	4	0.000002	19.294	8.497	34.0
70505	house has	4	0.000002	19.294	2.015	8.1
80278	house from	4	0.000002	19.294	0.690	2.8
90260	house ;	4	0.000002	19.294	0.413	1.7
82307	house for	4	0.000002	19.294	0.173	0.7
120579	house shall	3	0.000001	19.709	5.249	15.7
96089	house national	3	0.000001	19.709	2.940	8.8
105389	house is	3	0.000001	19.709	-0.742	-2.2
177819	house correspondent	2	0.000001	20.294	8.789	17.6
178456	house speaker	2	0.000001	20.294	7.557	15.1
130699	house staff	2	0.000001	20.294	6.482	13.0
205515	house encyclopedia	2	0.000001	20.294	6.439	12.9
166491	house near	2	0.000001	20.294	2.918	5.8
146131	house during	2	0.000001	20.294	1.470	2.9

Word pairs, with *house* on left side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
499	house of	274	0.000107	13.196	3.515	963.1
5928	house (	40	0.000016	15.972	2.559	102.4
3183	house ,	65	0.000025	15.271	0.751	48.8
77031	house arrest	4	0.000002	19.294	8.497	34.0
55523	house may	6	0.000002	18.709	3.154	18.9
177819	house correspondent	2	0.000001	20.294	8.789	17.6
120579	house shall	3	0.000001	19.709	5.249	15.7
178456	house speaker	2	0.000001	20.294	7.557	15.1
130699	house staff	2	0.000001	20.294	6.482	13.0
205515	house encyclopedia	2	0.000001	20.294	6.439	12.9
622825	house t'	1	0.000000	21.294	12.111	12.1
623903	house varied-depending	1	0.000000	21.294	12.111	12.1
12045	house in	22	0.000009	16.834	0.497	10.9
384777	house centipede	1	0.000000	21.294	10.111	10.1
706010	house searches	1	0.000000	21.294	9.111	9.1
457123	house wiring	1	0.000000	21.294	8.941	8.9
96089	house national	3	0.000001	19.709	2.940	8.8
380911	house carpenter	1	0.000000	21.294	8.526	8.5
340965	house chooses	1	0.000000	21.294	8.304	8.3
332198	house appropriations	1	0.000000	21.294	8.111	8.1
70505	house has	4	0.000002	19.294	2.015	8.1
248760	house finch	1	0.000000	21.294	7.789	7.8
480626	house dwellers	1	0.000000	21.294	7.205	7.2
664208	house originating	1	0.000000	21.294	6.982	7.0
633864	house shrine	1	0.000000	21.294	6.902	6.9

Word pairs, with *house* on right side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
637	the house	225	0.000088	13.480	2.474	556.7
7924	a house	31	0.000012	16.339	1.695	52.5
8671	. house	29	0.000011	16.436	0.219	6.4
11715	opera house	23	0.000009	16.770	9.080	208.8
19327	lower house	15	0.000006	17.387	7.167	107.5
19060	each house	15	0.000006	17.387	5.506	82.6
22523	white house	13	0.000005	17.593	6.510	84.6
34760	court house	9	0.000004	18.124	5.469	49.2
35512	and house	8	0.000003	18.294	-1.160	-9.3
53432	royal house	6	0.000002	18.709	5.674	34.0
64761	upper house	5	0.000002	18.972	6.389	31.9
62681	either house	5	0.000002	18.972	5.317	26.6
59445	state house	5	0.000002	18.972	3.322	16.6
63693	(house	5	0.000002	18.972	-0.441	-2.2
65857	, house	5	0.000002	18.972	-2.949	-14.7
84487	random house	4	0.000002	19.294	9.157	36.6
72133	hull house	4	0.000002	19.294	8.982	35.9
73936	customs house	4	0.000002	19.294	6.952	27.8
81491	country house	4	0.000002	19.294	4.140	16.6
81357	british house	4	0.000002	19.294	3.461	13.8
86882	his house	4	0.000002	19.294	0.807	3.2
84279	that house	4	0.000002	19.294	0.542	2.2
109679	ruling house	3	0.000001	19.709	7.321	22.0
118812	government house	3	0.000001	19.709	2.865	8.6
101702	under house	3	0.000001	19.709	2.731	8.2

Word pairs, with *house* on right side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
637	the house	225	0.000088	13.480	2.474	556.7
11715	opera house	23	0.000009	16.770	9.080	208.8
19327	lower house	15	0.000006	17.387	7.167	107.5
22523	white house	13	0.000005	17.593	6.510	84.6
19060	each house	15	0.000006	17.387	5.506	82.6
7924	a house	31	0.000012	16.339	1.695	52.5
34760	court house	9	0.000004	18.124	5.469	49.2
84487	random house	4	0.000002	19.294	9.157	36.6
72133	hull house	4	0.000002	19.294	8.982	35.9
53432	royal house	6	0.000002	18.709	5.674	34.0
64761	upper house	5	0.000002	18.972	6.389	31.9
73936	customs house	4	0.000002	19.294	6.952	27.8
62681	either house	5	0.000002	18.972	5.317	26.6
162587	carlton house	2	0.000001	20.294	11.526	23.1
109679	ruling house	3	0.000001	19.709	7.321	22.0
154063	somerset house	2	0.000001	20.294	9.652	19.3
162648	manor house	2	0.000001	20.294	9.411	18.8
59445	state house	5	0.000002	18.972	3.322	16.6
81491	country house	4	0.000002	19.294	4.140	16.6
81357	british house	4	0.000002	19.294	3.461	13.8
192997	neither house	2	0.000001	20.294	6.572	13.1
252306	music-publishing house	1	0.000000	21.294	12.111	12.1
262579	finlandia house	1	0.000000	21.294	12.111	12.1
275419	chief's house	1	0.000000	21.294	12.111	12.1
279889	tullie house	1	0.000000	21.294	12.111	12.1

3.9 French de

Words, sorted by frequency

rank	word	count	frequency	plog
1	,	595592	0.068	3.880
2	de	456678	0.052	4.263
3	.	376252	0.043	4.543
4	la	301251	0.034	4.864
5	et	228471	0.026	5.263
6	le	198547	0.023	5.465
7	les	193829	0.022	5.500
8	des	162899	0.019	5.751
9	à	158076	0.018	5.794
10	en	141333	0.016	5.956
11	du	112781	0.013	6.281
12	)	93263	0.011	6.555
13	(	93189	0.011	6.556
14	dans	73744	0.008	6.894
15	par	71346	0.008	6.942
16	une	63388	0.007	7.112
17	un	62680	0.007	7.129
18	!	58448	0.007	7.229
19	qui	55319	0.006	7.309
20	au	55022	0.006	7.317
21	est	53503	0.006	7.357
22	il	52046	0.006	7.397
23	pour	43163	0.005	7.667
24	plus	37451	0.004	7.872
25	que	36473	0.004	7.910

Word pairs, sorted by bigram frequency

rank	bigram	count	frequency	plog	MI	weighted MI
1	. .	92398	0.010492	6.575	2.511	232040.482
2	de la	82589	0.009378	6.737	2.391	197442.068
3	) ,	36538	0.004149	7.913	2.523	92168.874
4	, le	30075	0.003415	8.194	1.152	34634.721
5	à la	27628	0.003137	8.316	2.341	64688.161
6	. les	27473	0.003120	8.324	1.718	47209.126
7	, les	25779	0.002927	8.416	0.964	24849.392
8	, la	24622	0.002796	8.483	0.262	6438.885
9	. le	23396	0.002657	8.556	1.452	33969.446
10	. la	22957	0.002607	8.584	0.823	18896.382
11	, il	21894	0.002486	8.652	2.625	57476.213
12	, qui	21098	0.002396	8.705	2.484	52402.933
13	, et	20904	0.002374	8.719	0.424	8869.472
14	et de	20860	0.002369	8.722	0.804	16779.840
15	! ;	19794	0.002248	8.797	7.200	142509.452
16	, en	19203	0.002181	8.841	0.995	19102.392
17	. il	17997	0.002044	8.935	3.005	54081.882
18	) .	16909	0.001920	9.025	2.074	35061.948
19	. en	15769	0.001791	9.125	1.373	21653.130
20	et les	14794	0.001680	9.217	1.545	22857.744
21	dans le	14449	0.001641	9.251	3.108	44903.784
22	, de	14370	0.001632	9.259	-1.116	-16031.025
23	dans les	13924	0.001581	9.305	3.089	43011.843
24	et le	13371	0.001518	9.363	1.364	18244.296
25	dans la	13215	0.001501	9.380	2.377	31418.196

Word pairs, sorted by repelling bigram mutual information

rank	bigram	count	frequency	plog	MI	weighted MI
1728041	la de	1	0.000000	23.070	-13.943	-13.9
1425939	les la	1	0.000000	23.070	-12.707	-12.7
1142776	la des	1	0.000000	23.070	-12.456	-12.5
1774443	le et	1	0.000000	23.070	-12.342	-12.3
1145039	des et	1	0.000000	23.070	-12.057	-12.1
458967	de à	2	0.000000	22.070	-12.013	-24.0
1288153	le des	1	0.000000	23.070	-11.854	-11.9
528173	de en	2	0.000000	22.070	-11.851	-23.7
1826839	des les	1	0.000000	23.070	-11.820	-11.8
538057	en .	2	0.000000	22.070	-11.572	-23.1
1605348	du et	1	0.000000	23.070	-11.526	-11.5
310977	les .	3	0.000000	21.485	-11.442	-34.3
1280004	de il	1	0.000000	23.070	-11.410	-11.4
1198328	des en	1	0.000000	23.070	-11.364	-11.4
323008	la et	3	0.000000	21.485	-11.359	-34.1
1499628	en à	1	0.000000	23.070	-11.321	-11.3
769948	la dans	1	0.000000	23.070	-11.312	-11.3
1810844	et)	1	0.000000	23.070	-11.252	-11.3
1787976	une la	1	0.000000	23.070	-11.094	-11.1
802782	un la	1	0.000000	23.070	-11.078	-11.1
537141	(.	2	0.000000	22.070	-10.971	-21.9
339059	et et	3	0.000000	21.485	-10.960	-32.9
166757	de et	6	0.000001	20.485	-10.959	-65.8
527838	de dans	2	0.000000	22.070	-10.913	-21.8
1675721	pour .	1	0.000000	23.070	-10.860	-10.9

Word pairs, sorted by attracting bigram mutual information

rank	bigram	count	frequency	plog	MI	weighted MI
211182	médard chouart	4	0.000000	21.070	21.058	84.2
213948	intracoastal waterway	4	0.000000	21.070	21.058	84.2
223771	pardo bazán	4	0.000000	21.070	21.058	84.2
224614	iuliu maniu	4	0.000000	21.070	21.058	84.2
225238	ibl al-haytham	4	0.000000	21.070	21.058	84.2
225348	iasnaäa poliana	4	0.000000	21.070	21.058	84.2
234999	marja al-taqlid	4	0.000000	21.070	21.058	84.2
237094	nasjonal samling	4	0.000000	21.070	21.058	84.2
238568	gentis anglorum	4	0.000000	21.070	21.058	84.2
246271	l’afrika korps	4	0.000000	21.070	21.058	84.2
251958	calvo sotelo	4	0.000000	21.070	21.058	84.2
255633	mwene mutapa	4	0.000000	21.070	21.058	84.2
260794	dazai osamu	4	0.000000	21.070	21.058	84.2
262923	entamoeba histolytica	4	0.000000	21.070	21.058	84.2
175091	santissima annunziata	5	0.000001	20.748	20.736	103.7
181551	geheime staatspolizei	5	0.000001	20.748	20.736	103.7
194347	llano estacado	5	0.000001	20.748	20.736	103.7
196837	delirium tremens	5	0.000001	20.748	20.736	103.7
216412	revolutionibus orbium	4	0.000000	21.070	20.736	82.9
223179	karlovy vary	4	0.000000	21.070	20.736	82.9
225153	kryvyï rih	4	0.000000	21.070	20.736	82.9
225195	bronislava nijinska	4	0.000000	21.070	20.736	82.9
231359	d’amadou koumba	4	0.000000	21.070	20.736	82.9
231620	tupac amaru	4	0.000000	21.070	20.736	82.9
233420	gösta berling	4	0.000000	21.070	20.736	82.9

Word pairs, sorted by attracting bigram weighted mutual information

rank	bigram	count	frequency	plog	MI	weighted MI
1	. .	4	0.000000	6.575	2.511	232040.5
2	de la	4	0.000000	6.737	2.391	197442.1
15	! ;	4	0.000000	8.797	7.200	142509.5
3	) ,	4	0.000000	7.913	2.523	92168.9
5	à la	4	0.000000	8.316	2.341	64688.2
11	, il	4	0.000000	8.652	2.625	57476.2
17	. il	4	0.000000	8.935	3.005	54081.9
12	, qui	4	0.000000	8.705	2.484	52402.9
6	. les	4	0.000000	8.324	1.718	47209.1
21	dans le	4	0.000000	9.251	3.108	44903.8
23	dans les	4	0.000000	9.305	3.089	43011.8
18	) .	4	0.000000	9.025	2.074	35061.9
4	, le	4	0.000000	8.194	1.152	34634.7
9	. le	4	0.000000	8.556	1.452	33969.4
25	dans la	4	0.000000	9.380	2.377	31418.2
31	, mais	4	0.000000	9.892	3.201	29677.6
28	par les	4	0.000000	9.710	2.732	28722.9
129	guerre mondiale	4	0.000000	11.614	9.491	26660.3
49	il fut	4	0.000000	10.671	4.782	25825.4
82	au cours	4	0.000000	11.186	6.739	25471.6
42	la ville	4	0.000000	10.505	4.111	24911.1
7	, les	4	0.000000	8.416	0.964	24849.4
87	ainsi que	4	0.000000	11.224	6.501	23941.5
37	sur le	4	0.000000	10.255	3.192	23011.5
20	et les	4	0.000000	9.217	1.545	22857.7

Word pairs, sorted by bigram count

with *de*

rank	bigram	count	frequency	plog	MI	weighted MI
2	de la	82589	0.009378	6.737	2.391	197442.1
14	et de	20860	0.002369	8.722	0.804	16779.8
22	, de	14370	0.001632	9.259	-1.116	-16031.0
38	de son	6685	0.000759	10.363	2.077	13883.8
60	de ses	4578	0.000520	10.910	2.087	9553.2
78	. de	3839	0.000436	11.164	-2.357	-9049.4
79	de nombreux	3818	0.000434	11.172	3.975	15176.7
94	de sa	3508	0.000398	11.294	1.857	6515.5
107	plus de	3208	0.000364	11.423	0.712	2285.2
117	de cette	2982	0.000339	11.528	2.097	6254.4
123	partir de	2902	0.000330	11.567	3.548	10295.2
128	de nombreuses	2828	0.000321	11.605	3.973	11236.5
140	de ces	2687	0.000305	11.678	2.230	5992.2
145	nom de	2612	0.000297	11.719	2.861	7474.0
150	ou de	2555	0.000290	11.751	0.675	1725.1
151	près de	2551	0.000290	11.753	3.769	9614.0
160	de ce	2448	0.000278	11.813	1.528	3739.4
163	de leur	2416	0.000274	11.832	1.877	4534.4
171	partie de	2346	0.000266	11.874	2.591	6078.4
178	nombre de	2268	0.000258	11.923	3.259	7390.4
182	de plus	2236	0.000254	11.943	0.192	428.4
202	cours de	1991	0.000226	12.111	2.761	5496.2
211	lors de	1923	0.000218	12.161	3.276	6299.2
239	forme de	1716	0.000195	12.325	2.542	4362.9
267	de deux	1534	0.000174	12.487	1.136	1742.8

Word pairs, with *de* sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
2	de la	82589	0.009378	6.737	2.391	197442.1
14	et de	20860	0.002369	8.722	0.804	16779.8
79	de nombreux	3818	0.000434	11.172	3.975	15176.7
38	de son	6685	0.000759	10.363	2.077	13883.8
128	de nombreuses	2828	0.000321	11.605	3.973	11236.5
123	partir de	2902	0.000330	11.567	3.548	10295.2
151	près de	2551	0.000290	11.753	3.769	9614.0
60	de ses	4578	0.000520	10.910	2.087	9553.2
145	nom de	2612	0.000297	11.719	2.861	7474.0
178	nombre de	2268	0.000258	11.923	3.259	7390.4
94	de sa	3508	0.000398	11.294	1.857	6515.5
211	lors de	1923	0.000218	12.161	3.276	6299.2
117	de cette	2982	0.000339	11.528	2.097	6254.4
171	partie de	2346	0.000266	11.874	2.591	6078.4
140	de ces	2687	0.000305	11.678	2.230	5992.2
202	cours de	1991	0.000226	12.111	2.761	5496.2
315	l'université de	1397	0.000159	12.622	3.365	4701.3
163	de leur	2416	0.000274	11.832	1.877	4534.4
337	de l'empire	1322	0.000150	12.702	3.326	4397.4
239	forme de	1716	0.000195	12.325	2.542	4362.9
375	afin de	1209	0.000137	12.831	3.550	4291.4
319	celui de	1382	0.000157	12.638	3.093	4275.2
410	de fer	1132	0.000129	12.926	3.646	4126.9
342	celle de	1314	0.000149	12.710	2.958	3886.6
408	raison de	1144	0.000130	12.910	3.319	3796.7

Word pairs, with *de* on left side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
2	de la	82589	0.009378	6.737	2.391	197442.1
38	de son	6685	0.000759	10.363	2.077	13883.8
60	de ses	4578	0.000520	10.910	2.087	9553.2
79	de nombreux	3818	0.000434	11.172	3.975	15176.7
94	de sa	3508	0.000398	11.294	1.857	6515.5
117	de cette	2982	0.000339	11.528	2.097	6254.4
128	de nombreuses	2828	0.000321	11.605	3.973	11236.5
140	de ces	2687	0.000305	11.678	2.230	5992.2
160	de ce	2448	0.000278	11.813	1.528	3739.4
163	de leur	2416	0.000274	11.832	1.877	4534.4
182	de plus	2236	0.000254	11.943	0.192	428.4
267	de deux	1534	0.000174	12.487	1.136	1742.8
289	de leurs	1472	0.000167	12.547	2.168	3190.7
306	de france	1426	0.000162	12.592	1.789	2551.1
321	de se	1380	0.000157	12.640	-0.295	-407.8
337	de l’empire	1322	0.000150	12.702	3.326	4397.4
381	de paris	1195	0.000136	12.847	2.655	3173.2
394	de “	1165	0.000132	12.884	0.577	672.1
410	de fer	1132	0.000129	12.926	3.646	4126.9
442	de de	1030	0.000117	13.062	-4.535	-4670.8
449	de façon	1012	0.000115	13.087	3.658	3702.4
455	de plusieurs	1007	0.000114	13.094	1.588	1598.9
466	de même	989	0.000112	13.120	1.252	1238.2
478	de nouvelles	967	0.000110	13.153	3.383	3271.4
511	de guerre	922	0.000105	13.222	0.836	771.0

Word pairs, with *de* on left side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
2	de la	82589	0.009378	6.737	2.391	197442.1
79	de nombreux	3818	0.000434	11.172	3.975	15176.7
38	de son	6685	0.000759	10.363	2.077	13883.8
128	de nombreuses	2828	0.000321	11.605	3.973	11236.5
60	de ses	4578	0.000520	10.910	2.087	9553.2
94	de sa	3508	0.000398	11.294	1.857	6515.5
117	de cette	2982	0.000339	11.528	2.097	6254.4
140	de ces	2687	0.000305	11.678	2.230	5992.2
163	de leur	2416	0.000274	11.832	1.877	4534.4
337	de l'empire	1322	0.000150	12.702	3.326	4397.4
410	de fer	1132	0.000129	12.926	3.646	4126.9
160	de ce	2448	0.000278	11.813	1.528	3739.4
449	de façon	1012	0.000115	13.087	3.658	3702.4
478	de nouvelles	967	0.000110	13.153	3.383	3271.4
289	de leurs	1472	0.000167	12.547	2.168	3190.7
381	de paris	1195	0.000136	12.847	2.655	3173.2
546	de manière	874	0.000099	13.299	3.439	3005.3
520	de l'le	908	0.000103	13.244	2.896	2629.7
306	de france	1426	0.000162	12.592	1.789	2551.1
600	de l'ordre	805	0.000091	13.417	3.144	2530.9
729	de nouveaux	697	0.000079	13.625	3.487	2430.5
526	de l'est	900	0.000102	13.256	2.573	2315.3
653	de l'homme	755	0.000086	13.510	3.017	2277.9
609	de l'eau	797	0.000090	13.432	2.858	2277.7
697	de l'europe	721	0.000082	13.576	3.112	2243.8

Word pairs, with *de* on right side, sorted by bigram count

rank	bigram	count	frequency	plog	MI	weighted MI
14	et de	20860	0.002369	8.722	0.804	16779.8
22	, de	14370	0.001632	9.259	-1.116	-16031.0
78	. de	3839	0.000436	11.164	-2.357	-9049.4
107	plus de	3208	0.000364	11.423	0.712	2285.2
123	partir de	2902	0.000330	11.567	3.548	10295.2
145	nom de	2612	0.000297	11.719	2.861	7474.0
150	ou de	2555	0.000290	11.751	0.675	1725.1
151	près de	2551	0.000290	11.753	3.769	9614.0
171	partie de	2346	0.000266	11.874	2.591	6078.4
178	nombre de	2268	0.000258	11.923	3.259	7390.4
202	cours de	1991	0.000226	12.111	2.761	5496.2
211	lors de	1923	0.000218	12.161	3.276	6299.2
239	forme de	1716	0.000195	12.325	2.542	4362.9
276	nord de	1503	0.000171	12.517	1.950	2930.8
283	est de	1481	0.000168	12.538	-0.917	-1358.6
310	sud de	1412	0.000160	12.607	2.104	2971.0
313	fin de	1400	0.000159	12.619	2.245	3142.3
315	l'université de	1397	0.000159	12.622	3.365	4701.3
319	celui de	1382	0.000157	12.638	3.093	4275.2
324	) de	1367	0.000155	12.653	-1.835	-2507.9
325	ville de	1366	0.000155	12.654	1.361	1859.4
339	roi de	1321	0.000150	12.703	2.285	3018.3
342	celle de	1314	0.000149	12.710	2.958	3886.6
354	centre de	1280	0.000145	12.748	2.471	3162.9
375	afin de	1209	0.000137	12.831	3.550	4291.4

Word pairs, with *de* on right side, sorted by Weighted Mutual Information

rank	bigram	count	frequency	plog	MI	weighted MI
14	et de	20860	0.002369	8.722	0.804	16779.8
123	partir de	2902	0.000330	11.567	3.548	10295.2
151	près de	2551	0.000290	11.753	3.769	9614.0
145	nom de	2612	0.000297	11.719	2.861	7474.0
178	nombre de	2268	0.000258	11.923	3.259	7390.4
211	lors de	1923	0.000218	12.161	3.276	6299.2
171	partie de	2346	0.000266	11.874	2.591	6078.4
202	cours de	1991	0.000226	12.111	2.761	5496.2
315	l'université de	1397	0.000159	12.622	3.365	4701.3
239	forme de	1716	0.000195	12.325	2.542	4362.9
375	afin de	1209	0.000137	12.831	3.550	4291.4
319	celui de	1382	0.000157	12.638	3.093	4275.2
342	celle de	1314	0.000149	12.710	2.958	3886.6
408	raison de	1144	0.000130	12.910	3.319	3796.7
398	traité de	1159	0.000132	12.892	3.261	3779.9
508	série de	925	0.000105	13.217	3.462	3202.0
354	centre de	1280	0.000145	12.748	2.471	3162.9
313	fin de	1400	0.000159	12.619	2.245	3142.3
339	roi de	1321	0.000150	12.703	2.285	3018.3
519	autour de	908	0.000103	13.244	3.306	3001.9
310	sud de	1412	0.000160	12.607	2.104	2971.0
390	mort de	1168	0.000133	12.880	2.510	2932.2
276	nord de	1503	0.000171	12.517	1.950	2930.8
409	avant de	1142	0.000130	12.913	2.321	2650.4
694	types de	726	0.000082	13.566	3.590	2606.5

word	phonemes	unigram plog	bigram plog	MI	average unigram plog	average bigram plog
A	# AH0 #	6.14	9.64	-3.50	3.07	4.82
A'S	# EY1 Z #	13.60	13.45	0.15	4.53	4.48
AACHEN	# AA1 K AH0 N #	21.16	17.20	3.96	4.23	3.44
AAMODT	# AA1 M AH0 T #	21.82	19.15	2.67	4.36	3.83
AARDVARK	# AA1 R D V AA1 R K #	39.70	36.67	3.04	4.96	4.58
AARON	# EH1 R AH0 N #	20.51	15.50	5.01	4.10	3.10
AARON'S	# EH1 R AH0 N Z #	25.79	17.35	8.44	4.30	2.89
ABABA	# AA1 B AH0 B AH0 #	27.93	26.19	1.73	4.65	4.37
ABACK	# AH0 B AE1 K #	22.59	20.24	2.35	4.52	4.05
ABACO	# AE1 B AH0 K OW1 #	29.38	27.45	1.93	4.90	4.57
ABACUS	# AE1 B AH0 K AH0 S #	31.07	26.55	4.53	4.44	3.79
ABAD	# AH0 B AA1 D #	22.98	20.63	2.35	4.60	4.13
ABALKIN	# AH0 B AA1 L K IH0 N #	36.62	34.69	1.93	4.58	4.34
ABALONE	# AE1 B AH0 L OW1 N IY0 #	39.18	30.48	8.70	4.90	3.81
ABANDON	# AH0 B AE1 N D AH0 N #	35.30	26.14	9.16	4.41	3.27
ABANDONED	# AH0 B AE1 N D AH0 N D #	40.28	28.64	11.63	4.48	3.18
ABANDONING	# AH0 B AE1 N D AH0 N IH0 NG #	46.87	30.87	16.00	4.69	3.09
ABANDONMENT	# AH0 B AE1 N D AH0 N M AH0 N T #	53.27	40.32	12.96	4.44	3.36
ABANDONS	# AH0 B AE1 N D AH0 N Z #	40.58	27.99	12.59	4.51	3.11
ABANTO	# AH0 B AE1 N T OW0 #	34.11	26.52	7.58	4.87	3.79
ABATE	# AH0 B EY1 T #	22.61	19.73	2.89	4.52	3.95
ZUKIN	# Z UW1 K IH0 N #	28.59	31.61	-3.03	4.76	5.27
ZUKOWSKI	# Z AH0 K AO1 F S K IY0 #	44.35	42.75	1.61	4.93	4.75
ZULAUF	# Z UW1 L AW0 F #	38.33	41.99	-3.66	6.39	7.00
ZULU	# Z UW1 L UW1 #	26.33	29.53	-3.20	5.27	5.91
ZULUS	# Z UW1 L UW0 Z #	33.83	33.61	0.22	5.64	5.60
ZURICH	# Z UH1 R IH0 K #	30.76	31.35	-0.59	5.13	5.23
ZURICH'S	# Z UH1 R IH0 K S #	35.19	33.22	1.97	5.03	4.75
ZURN	# Z ER1 N #	19.33	21.45	-2.12	4.83	5.36
ZWEIBEL	# Z W AY1 B AH0 L #	35.51	33.09	2.43	5.07	4.73
ZWEIG	# Z W AY1 G #	27.60	30.86	-3.26	5.52	6.17
ZWETCHKENBAUM	# Z W EH1 CH K AH0 N B AA0 M #	60.93	59.37	1.56	5.54	5.40
ZWICK	# Z W IH1 K #	24.99	26.07	-1.08	5.00	5.21
ZYDECO	# Z IH1 D AH0 K OW1 #	33.81	36.86	-3.05	4.83	5.27
ZYDECO(3)	# Z AY1 D AH0 K OW1 #	34.61	36.13	-1.52	4.94	5.16
ZYGOTE	# Z AY1 G OW0 T #	32.67	35.74	-3.06	5.45	5.96
ZYMAN	# Z AY1 M AH0 N #	27.61	26.45	1.16	4.60	4.41
{BRACE	# B R EY1 S #	23.22	18.04	5.18	4.64	3.61
{LEFT-BRACE	# L EH1 F T B R EY1 S #	44.06	42.18	1.88	4.90	4.69
}CLOSE-BRACE	# K L OW1 Z B R EY1 S #	44.66	38.07	6.59	4.96	4.23
}RIGHT-BRACE	# R AY1 T B R EY1 S #	38.82	34.22	4.60	4.85	4.28

word	phonemes	unigram plog	bigram plog	MI	average unigram plog	average bigram plog
A	# AH0 #	6.14	9.64	-3.50	3.07	4.82
AN	# AH0 N #	10.38	9.42	0.96	3.46	3.14
TO(3)	# T AH0 #	10.51	12.23	-1.72	3.50	4.08
LE	# L AH0 #	10.72	12.42	-1.69	3.57	4.14
DU	# D AH0 #	11.11	12.01	-0.89	3.70	4.00
DE(3)	# D AH0 #	11.11	12.01	-0.89	3.70	4.00
CAN	# K AH0 N #	15.16	10.48	4.68	3.79	2.62
EH	# EH1 #	7.63	17.15	-9.53	3.81	8.58
TO	# T IH0 #	11.46	20.04	-8.58	3.82	6.68
IT	# IH0 T #	11.46	11.91	-0.45	3.82	3.97
AND	# AH0 N D #	15.35	11.92	3.43	3.84	2.98
OR	# ER0 #	7.75	11.67	-3.92	3.87	5.83
ER	# ER0 #	7.75	11.67	-3.92	3.87	5.83
ARE	# ER0 #	7.75	11.67	-3.92	3.87	5.83
INTO(3)	# IH0 N T AH0 #	19.75	18.41	1.34	3.95	3.68
N	# EH1 N #	11.87	10.98	0.88	3.96	3.66
EN	# EH1 N #	11.87	10.98	0.88	3.96	3.66
N.	# EH1 N #	11.87	10.98	0.88	3.96	3.66
ITS	# IH0 T S #	15.89	14.96	0.94	3.97	3.74
IT'S	# IH0 T S #	15.89	14.96	0.94	3.97	3.74
ET	# EH1 T #	12.00	12.56	-0.56	4.00	4.19
YAOBANG	# Y AW1 B AE0 NG #	40.16	36.28	3.88	6.69	6.05
WYETH	# W AY1 EH0 TH #	33.52	29.83	3.69	6.70	5.97
REGIME	# R EY0 ZH IY1 M #	40.36	32.35	8.01	6.73	5.39
LITHGOW	# L IH1 TH G AW0 #	40.51	37.87	2.64	6.75	6.31
WOJCIECH	# V OY1 CH EH0 K #	40.53	31.77	8.76	6.75	5.30
VIRTUE	# V ER1 CH UW0 #	33.78	24.28	9.50	6.76	4.86
THYROID	# TH AY1 R OY0 D #	40.65	33.31	7.34	6.77	5.55
HAUPPAUGE	# HH AW1 P AO0 JH #	40.94	36.90	4.03	6.82	6.15
CESARE	# CH EY0 Z AA1 R EY0 #	47.88	41.49	6.39	6.84	5.93
TOYOO	# T OY0 UW1 #	27.43	27.62	-0.19	6.86	6.91
GIRAUD	# ZH AY0 R OW1 #	34.33	32.64	1.69	6.87	6.53
EURASIA	# Y UH0 R EY1 ZH AH0 #	48.09	31.49	16.59	6.87	4.50
BOURGEOIS	# B UH1 R ZH W AA0 #	48.10	41.78	6.32	6.87	5.97
QURESHEY	# K UH0 R EY1 SH EY0 #	48.35	36.02	12.33	6.91	5.15
CEAUSESCU	# CH AW0 CH EH1 S K Y UW0 #	62.61	46.62	15.99	6.96	5.18
GEOID	# JH IY1 OY0 D #	34.88	26.75	8.13	6.98	5.35
PEUGEOT	# P Y UW0 ZH OW1 #	42.30	33.56	8.74	7.05	5.59
THOU	# DH AW1 #	21.34	23.70	-2.36	7.11	7.90
THURGOOD	# TH ER1 G UH0 D #	42.67	35.21	7.46	7.11	5.87
CHENOWETH	# CH EH1 N AW0 EH0 TH #	50.38	41.32	9.06	7.20	5.90

word	phonemes	unigram plog	bigram plog	MI	average unigram plog	average bigram plog
STATIONS	# S T EY1 SH AH0 N Z #	37.58	20.21	17.38	4.70	2.53
STATIONS’	# S T EY1 SH AH0 N Z #	37.58	20.21	17.38	4.70	2.53
STATION’S	# S T EY1 SH AH0 N Z #	37.58	20.21	17.38	4.70	2.53
STATIONING	# S T EY1 SH AH0 N IH0 NG #	43.87	23.08	20.79	4.87	2.56
PARENTING	# P EH1 R AH0 N T IH0 NG #	42.07	23.30	18.77	4.67	2.59
CORRELATIONS	# K AO1 R AH0 L EY1 SH AH0 N Z #	53.61	28.65	24.96	4.87	2.60
STATIONED	# S T EY1 SH AH0 N D #	37.28	20.86	16.42	4.66	2.61
RATIONING	# R EY1 SH AH0 N IH0 NG #	39.67	20.87	18.80	4.96	2.61
HANDING	# HH AE1 N D IH0 NG #	35.81	18.31	17.50	5.12	2.62
CAN	# K AH0 N #	15.16	10.48	4.68	3.79	2.62
WARREN’S	# W AO1 R AH0 N Z #	34.05	18.35	15.70	4.86	2.62
STATION	# S T EY1 SH AH0 N #	32.31	18.36	13.95	4.62	2.62
CONTESTING	# K AH0 N T EH1 S T IH0 NG #	45.44	26.34	19.11	4.54	2.63
CONTENDING	# K AH0 N T EH1 N D IH0 NG #	45.86	26.38	19.48	4.59	2.64
STRANDING	# S T R AE1 N D IH0 NG #	42.06	23.76	18.30	4.67	2.64
STATING	# S T EY1 T IH0 NG #	33.06	18.53	14.54	4.72	2.65
WARRING	# W AO1 R IH0 NG #	32.05	15.90	16.16	5.34	2.65
RATIONED	# R EY1 SH AH0 N D #	33.07	18.64	14.43	4.72	2.66
CONTRASTING	# K AH0 N T R AE1 S T IH0 NG #	50.29	29.30	20.99	4.57	2.66
BANDING	# B AE1 N D IH0 NG #	34.53	18.70	15.83	4.93	2.67
RANTING	# R AE1 N T IH0 NG #	32.65	18.71	13.94	4.66	2.67
AYR	# EY1 R #	12.92	21.53	-8.61	4.31	7.18
MUI	# M UW1 IH0 #	19.59	29.08	-9.49	4.90	7.27
AER	# EY1 IY1 AA1 R #	25.60	36.49	-10.89	5.12	7.30
THY	# DH AY1 #	19.53	22.24	-2.71	6.51	7.41
ARROYO	# ER0 OY1 OW0 #	25.32	30.03	-4.71	6.33	7.51
THEE	# DH IY1 #	19.58	22.53	-2.96	6.53	7.51
OAHU	# OW1 AA1 HH UW0 #	31.44	38.17	-6.73	6.29	7.63
DES	# D IH1 #	12.90	22.90	-10.00	4.30	7.63
ARAU	# AH0 R AW1 #	19.18	30.88	-11.70	4.80	7.72
ERR	# ER1 #	9.81	15.56	-5.75	4.91	7.78
OY	# OY1 #	11.96	15.70	-3.74	5.98	7.85
YE	# Y EH1 #	15.36	23.55	-8.19	5.12	7.85
THOU	# DH AW1 #	21.34	23.70	-2.36	7.11	7.90
OOH	# UW1 #	9.28	16.02	-6.75	4.64	8.01
ZSA	# ZH AA1 #	18.74	24.07	-5.33	6.25	8.02
UH	# AH1 #	9.11	16.19	-7.08	4.55	8.10
YEAH	# Y AE1 #	15.61	25.73	-10.11	5.20	8.58
EH	# EH1 #	7.63	17.15	-9.53	3.81	8.58
AI	# EY1 AY1 #	14.97	26.06	-11.09	4.99	8.69
THE	# DH AH1 #	19.91	26.15	-6.24	6.64	8.72

3.9.1 The problem of sparse data

The problem posed by sparse data is how to treat all the structures that occur rarely or not at all in the training data. Thus far, we have been using what are known as Maximum Likelihood Estimates (MLE) in our models.[20] Using MLE, the probability assigned to structure a, $p(a) = Count(a)/|S|$ is essentially its frequency. This approach provides the tightest fit between the parameters (i.e. probability estimates) in a model and the data with which it is trained. Consequently, any structures (phones, n-grams, whatever) that are not observed in the training data will be assigned zero probability, and that could be interpreted as a claim of grammatical impossibility, which is generally an imprudent leap of logic to take. It can be the case, however, that the missing structures are accidental gaps in the training data. When a model erroneously treats an accidental gap as a systematic gap the model is said to have *over-fit* the training data.

If the goal is the construction of a generative model, MLE probabilities are usually avoided because they yield models that are 'brittle' in the sense that the occurrence of a zero-probability element in a form nullifies all other distinctions (i.e. any pair of words containing zero probability elements have the same probability, zero, regardless of any other distinctions between them). This problem has been extensively studied in statistical natural language processing, and it has been approached with a wide range of sophisticated solutions that go by the general name of *smoothing* techniques.[21]

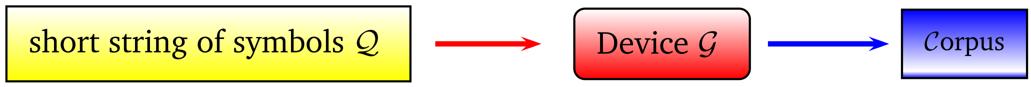
One of the most basic smoothing strategies is to use Laplace's Law in a scheme that adds one to all counts by initializing each count to one when computing frequencies. This is is a specific instance of a more general strategy of adding λ, called Lidstone's Law:

$$p(s) = \frac{Count(s) + \lambda}{N + B\lambda}, \tag{3.9}$$

where N is the total number of instances structures like s, and B is the number of possible kinds of structures like s. When $\lambda = 0$ this formula is simply the maximum likelihood estimator; this gives the best fit for the training data but reserves no probability for unseen events. When $\lambda = 1$ we are using what is usually referred to as Laplace's law, which corresponds conceptually to a uniform Bayesian prior over the possible structures. When $\lambda = 1/2$ we are using what is usually called the Jeffreys-Perks Law (though Perks more strongly advocated $\lambda = 1/|T|$ where T is the set of types). The value $\lambda = 1/2$ is also referred to as Expected Likelihood Estimation (ELE) and is the most commonly used fixed value for λ in language modeling. There are many strategies for calculating optimal values for λ in given contexts and, more generally, many other strategies for calculating the amount of probability to reserve for unseen events (see Manning and Schütze ([**?**, ch6]) for an overview and [**?**] for a thorough discussion of the development of many of

[20] *Probability* and *likelihood* are terms used in quite different ways. Here is how to keep them separate: a particular probability distribution assigned to a set is a function that takes something as its input, and gives a non-negative real (≤ 1) as its output. In the cases we are interested in, that input typically contains something symbolic (like the symbol *k*, say) and a distribution, a set of numbers, which either form a distribution or are used inside a mathematical expression that forms a distribution over the sample set. Thus it takes the form: $p(s, \lambda)$, where λ is a continuous parameter, or more often a set of continuous parameters. If we hold λ constant and let s vary, we have what we usually call a distribution; if we hold s fixed and let λ vary, we have a likeliness function. What value of λ will give the largest value of this function for a given symbolic string? The answer is that string's *maximum likelihood* specification.

[21] smoothing



these ideas). In our general presentation of models in the sections that follow we will use MLE probabilities. However, whenever we compare alternative models we will use ELE $\lambda = 1/2$ in smoothing the probabilities. Smoothing is useful in comparing alternative models to evaluate not only their ability to fit the data but also their tendency to over-fit the data. In §**??** we will discuss an alternative to smoothing whereby minimization of model complexity is used to avoid over-fitting.

3.10 Grammars and data description

We are going to use a style of thinking about data that is derived from a long line of people who have thought about computation, probability, and what it means to explain things. Among the people prominent on this list are Alan Turing, perhaps Rudolf Carnap, Ray Solomonoff, and Jorma Rissanen. This section is not intended to be understood right away: I expect that some of it will be reasonably clear, but some of it will not really make sense until we have explored more ideas of information theory. Still, this is here to give you a sense of where we are headed.

Let us suppose for a moment that our set of data $\mathcal{C}$ (our *corpus*) is from one particular language—English, let us say.

We will think of the problem that we face as that of providing a good analysis of data $\mathcal{C}$. To do that, we need to make clear what the words *analysis* and *good* mean in this context.

We will apply a dynamic and computational interpretation: an analysis of a set of data $\mathcal{C}$ is a device $\mathcal{G}$ that takes a string of symbols $\mathcal{Q}$ as its input and produces $\mathcal{C}$ as its output. We will explain in just a moment what the intuition is that lies behind this "$\mathcal{Q}$".

And we have three expectations:

1. first, we want the input $\mathcal{Q}$ to be as small as possible;
2. second, we want the device $\mathcal{G}$ to be as simple as possible; and
3. third, when we want to analyze two sets of data (two different *corpora*) from the same language, we want the device $\mathcal{G}$ to change very little or not at all.

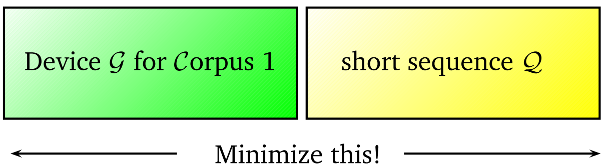
Everything else should follow from these initial desiderata.

Here are some points that we take to be natural inferences from this set of expections:

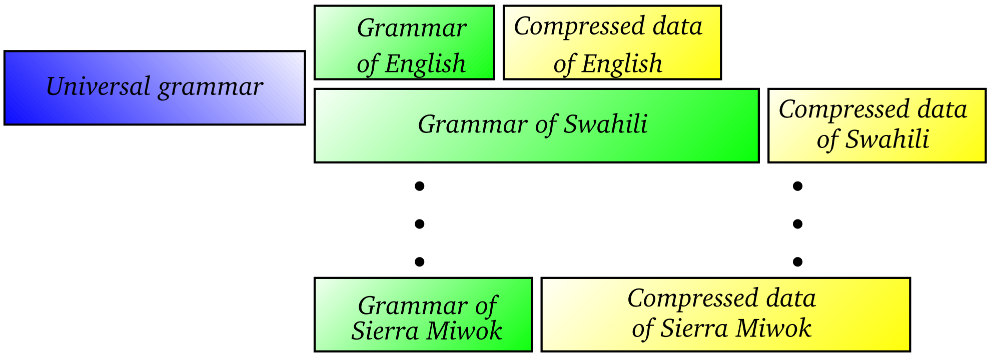
1. To make our task clearer (and perhaps simpler), we can insist that the input $\mathcal{Q}$ will take the form of a string of binary digits (0s and 1s), and then we measure the size of $\mathcal{Q}$ as simply its length. The length of Q will normally be less than the length of the corpus (in a sense that will become much clearer as we go along). We will eventually see that it makes sense to call $\mathcal{Q}$ the *compressed* version of the corpus $\mathcal{C}$orpus, and then we say that the length of $\mathcal{Q}$ is *the compressed length of the corpus, given* $\mathcal{G}$.

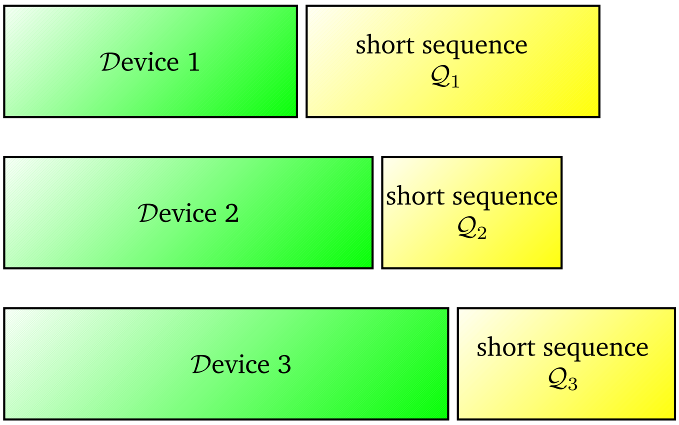
2. As a first approximation, we will take the device $\mathcal{G}$ to be a *program* in some chosen programming language, and its complexity to be based on the number of symbols, or characters, used in the program: in particular, it will be the number of characters multiplied by a weighting factor λ—and λ is there because for the moment we declared our $\mathcal{Q}$ would be expressed in binary (symbols = {0,1}), whereas in programming languages we have a larger alphabet. There will be an advantage to using a very austere programming style: there is a disadvantage to calling a variable *MaximumStringLength* and an advantage to calling it simply *g*, since we save 18 letters that way—but this is not a big deal. (We can imagine a preprocessor that tokenizes the program, and assigns maximally short variable names, and then use the output of the preprocessor for a length computation.)

3. We will find a way to express both the length of $\mathcal{Q}$ and the complexity of $\mathcal{G}$ in the same units (so that λ then equals 1), and then we can express our desire to make $\mathcal{Q}$ short and to make $\mathcal{G}$ simple as a desire to make the sum of the length of $\mathcal{Q}$ + complexity of $\mathcal{G}$ as small as possible, which we can represent visually as trying to keep minimize the width of these two blocks together:



4. For any given corpus $\mathcal{C}$, there are many possible analyses (where an analysis is a device $\mathcal{D}$ and its corresponding $\mathcal{Q}$), and they are not all equally good from the point of view of minimizing the sum of the two quantitites. You can see in the figure below that the second analysis is the best (the total width is the least) of the three.





5. Eventually we will think of our "device" as no more and no less than the *grammar* of the language—and that is the reason we chose to use the letter $\mathcal{G}$ as the symbol (rather than "D", for example).

6. If our data comes from two or more languages, then the device $\mathcal{G}$ is likely to split itself into two layers. One layer contains information that is relevant for both (or all) languages, and the other layer will partition into grammars for each language. We may refer to the part that is relevant for all languages as *universal grammar*. It can also be thought of as a compiler which defines the language in which individual grammars are written. So we will change our description slightly, and in the following figure, we write "grammar of English" rather than "Device 1," and so forth. But just as before, our over-all goal is to minimize the total area of the rectangles in the figure.

These remarks might sound abstract and even cryptic now, but they will guide a lot of what we do, and eventually they will seem intuitively clear, and reasonable.

3.11 Shannon information

Just some reminders about logarithms: Natural logs, base 10 logs. $ln(x)$ is the natural log (base e) of x. $y = e^{lny}$ by definition. What's y's base 2 log? Since $e = 2^{log_2 e}$ it follows that $y = (2^{log_2 e})^{ln(y)}$, which is $2^{log_2(e)ln(y)}$; so $log_2(y) = log_2(e)ln(y)$. This illustrates the more general fact that changing the base of a logarithm consistly just changes our numbers by a constant multiplicative factor. There's only one place where we really care about a special property of natural logs that log_2 do not have, and that's the fact that $ln(x) \leq x - 1$, which depends on the derivative of $ln(x)$ being 1 at $x = 1$.[22]

Now here is a trivial sort of algebraic manipulation that may not be totally obvious the first time you see it. We have been thinking of the probability as the product of the individual segments as we iterate through a string; but as long as we are considering unigram probabilities, independent of context, we could also compute the same value by taking the product of each of the items in the alphabet that produced the sequence, raising each letter in the alphabet to the power of how many times it occurred in the sequence:

In the expressions below, we use l to represent a variable that is defined over the alphabet (think l for $letter$)

$$\prod_{i=1}^{i=len(string)} S[i] = \prod_{l \in lexicon} l^{count_S(l)}. \tag{3.10}$$

$$logprob(S) = \sum_{lexicon} count_S(l) logprob(l). \tag{3.11}$$

$$plog(S) = \sum_{lexicon} count_S(l) plog(l). \tag{3.12}$$

If we divide through by the length of our string, we get the average which is Shannon's entropy:

$$entropy(S) = \sum_{lexicon} freq_S(l) plog(l). \tag{3.13}$$

since the count divided by the total count is the frequency. In short: the *entropy* of a message is its *average plog*. The term is generally employed when we are considering a large sample from some source.

[22]What is the slope of $log(x)$ when the base is something other than e?

Notice that both parts of the expression that we sum here involve probabilities or frequencies of each letter in our alphabet. Each involve a distribution, and conceptually these two distributions can be separated and made distinct. That is what we do next.

Cross entropy: where we keep the empirical frequencies, but vary the distribution whose plog we use to compute the entropy. This is the "cross-entropy" of one distribution to the other (but not symmetrical!). Entropy, or self-entropy, is always smaller than cross-entropy.

$$\sum_x p(x) ln \frac{q(x)}{p(x)} \leq \sum_x p(x)(1 - \frac{q(x)}{p(x)}) \tag{3.14}$$

Why? Look at the plot of $ln(x)$, and compute its first and second derivatives, and its value at (1,0).

$$= \sum_x p(x) - \sum_x p(x) \frac{q(x)}{p(x)} = 1 - 1 = 0. \tag{3.15}$$

So $\sum_x p(x)ln(\frac{q(x)}{p(x)} \leq 0$, which is to say, the cross-entropy always exceeds the entropy that isn't cross, when we use natural logs as our base.[23] But we can maintain the inequality when we switch to base 2 logs (which is what we use with plogs), since it just amounts to multiplying both sides by a constant. First we get:

$$\sum_x p(x) ln\, q(x) \leq \sum_x p(x) ln\, p(x) \tag{3.16}$$

and then we multiply by -1:

$$\sum_x p(x) plog p(x) \leq \sum_x p(x) plog\, q(x) \tag{3.17}$$

The Kullback-Leibler divergence $D_{KL}(p, q)$ is defined as [24]

$$\sum_x p(x)\, ln\, \frac{p(x)}{q(x)} \tag{3.18}$$

You see that it's the difference between the cross-entropy and the self-entropy—pay careful attention to the *absence* of a minus before the sum.

As we will see more clearly next class, given a distribution q(x) over an alphabet Σ, we can always construct an encoding of Σ (which is map into $\{0, 1\}$ with the prefix property) in which each symbol is encoded by a string whose length is no longer than the plog of that symbol's

[23] Cross-entropy $\geq$ (self-)entropy: always
[24] KL divergence

property (well, we may have to round up to get an integral number of bits, but eventually we can even get away from that restriction, hard as it may be to believe.)

An encoding has the prefix property iff there are no two $x, y \in \Sigma$ such that the encoding of x is the encoding of y followed by something else. Examples of encoding that does not have the prefix property: $a \to 1$; $b \to 11$; $c \to 01$. Given an encoding 111, we don't know whether it is an encoding of aaa, ab, or ba. We only want encoding systems with the prefix property, because they parse themselves (so to speak) as we scan them from left to right, and if we already know the encoding system, of course! This turns out to be an important consideration.

key words: entropy, cross-entropy, self-entropy, KL-divergence. Language ID.

3.12 Letters: Transitional phone (or letter) models

NB: I will use the terms "letter" and "phone" interchangeably here. Let's explore transitional letter models by seeing if we can write an algorithm that will identify the language in which a text is written, based on letter frequencies.

We are given a text T. We will assume that all of our texts are encoded in (some) standard Unicode, and we call that alphabet Σ, so $T \in \Sigma^*$. We are also given a set of texts from several known languages, which we hope are reasonably representative of their respective languages. We will consider three ways to assign probabilities to strings in Σ^*, and ways to draw the respective parameters from those sample texts.

In the terms used originally by Shannon, these would be 0-order, 1st order, and 2nd order models. Unfortunately, terminology has changed over time!

By a 0-order model, Shannon meant a model that assigns a uniform distribution over all of the symbols that are in the alphabet of the model. The usual way of interpreting that in this context is to say that we infer the alphabet used by a language from our sample: it is the smallest set of symbols Σ such that $S \in \Sigma^*$; and then we assign a uniform distribution over this alphabet. This choice will assign a probability of zero to any string containing one or more symbols not in the original sample.

By a 1st order model, Shannon meant a model in which the probability of a symbol pr_U (where 'U' stands for 'unigram') is taken to be its frequency in the sample, and the probability of a string is equal to the product of the probabilities of the symbols. (Note that we have to assume a distribution over string length as well, which is typically done by assuming the existence of a special symbol that only appears at the end of strings. We will return to this. For now, we will assume that there is a function $pr_l()$ which is a distribution over the positive integers, and which assigns a probability that a string will be of a given length.)[25]

$$p(s) = pr_l(|s|) \prod_{i=1}^{|s|} pr_U(s[i]) \tag{3.19}$$

But instead of calling this a 1st order model, we will call this a *unigram* model, since many writers today use the term "1st order model" to mean something else.

By a 2nd order model, Shannon meant a model much like the unigram model, but in which the probablity of a symbol was conditioned by the preceding symbol. We will call this a *bigram* model. (Many writers today call this a 1st order Markov model.)

Shannon, in "The mathematical theory of communication," gave three approximations of English:

Zero-order approximation: XFOML RXKHRJFFJUJ ALPWXFWJXYJ FFJEYVJCQSGHYD QPAAMKBZAACIBZLKJQD

First-order approximation: OCRO HLO RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL

Second-order approximation: ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

Third-order approximation: IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE

> We'll get to word-based models shortly, but I'll share with you Shannon's approximations of English using a word-based model:

[25] This kind of model was explored by early cryptographers, notably al-Kindi, who lived from 801-873, and who worked under the Abbasid caliphs in Baghdad, as well as al-Khwarizmi, c. 780-850. Al-kindi wrote (I have taken this from http://www.muslimheritage.com/topics/default.cfm?articleID=372, who quotes Singh, *The Code Book*):

> One way to solve an encrypted message, if we know its language, is to find a different plaintext of the same language long enough to fill one sheet or so, and then we count the occurrences of each letter. We call the most frequently occurring letter the 'first', the next most occurring letter the 'second', the following most occurring the 'third', and so on, until we account for all the different letters in the plaintext sample....
>
> Then we look at the cipher text we want to solve and we also classify its symbols. We find the most occurring symbol and change it to the form of the 'first' letter of the plaintext sample, the next most common symbol is changed to the form of the 'second' letter, and so on, until we account for all symbols of the cryptogram we want to solve.

rank	orthography	phonemes	log_1	$averageplog_1$
1	a	ə	6.23	3.11
2	an	ə n	10.33	3.44
3	to	t ə	10.40	3.47
4	and	ə n d	15.18	3.80
5	eh	ɛ́	6.23	3.88
6	the	ð ə	11.63	3.88
7	can	k ə n	15.60	3.90
8	an	ǽ n	11.72	3.91
9	Ann	ǽ n	11.72	3.91
10	in	í n	11.72	3.91
63195	bourgeois	b ʌ̆ r ž w á	50.44	7.21
63196	Ceausescu	č ɔ̆ č ɛ́ s k ŭ	64.86	7.21
63197	Peugeot	p y ŭ ž ó	43.34	7.22
63198	Giraud	ž ay̆ r ó	36.19	7.24
63199	Godoy	g á d oy̆	36.35	7.27
63200	geoid	ǰ í ɔy̆ d	37.00	7.40
63201	Cesare	č ĕ z á r ĕ	51.80	7.40
63202	Thurgood	θ ɚ́ g ʌ̆ d	44.86	7.47
63203	Chenoweth	č ɛ́ n ɔ̆ w ɛ̆ θ	52.46	7.49
63204	Qureshey	k ə r é š ĕ	52.77	7.54

Tab. 3.9: Top and bottom of English word list, based solely on unigram frequencies.

> First-order word approximation: REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE
>
> Second-order word approximation THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

[Some of this material, especially the tables, is from a paper by Jason Riggle and John Goldsmith, Information theoretical approaches to phonological structure: the case of vowel harmony.]

French oral vowels

Height	Vowel	example	Vowel	example	Vowel	example
	Front unrounded		Front rounded		Back	
High	i	vie	y	du	u	tout
Mid: tense	e	blé	ö	peu	o	mot
Mid: lax	ɛ	tête	œ	peur	ɔ	donne
Low:					a	plat

rank	phoneme	frequency	plog
1	#	0.20	2.30
2	ə	0.066	3.92
3	n	0.058	4.10
4	t	0.056	4.17
5	s	0.041	4.61
6	r	0.040	4.76
7	d	0.037	4.85
8	l	0.035	4.94
9	k	0.026	5.27
10	ǽ	0.025	5.31
45	ɔý	0.000 78	10.32
46	æ̆	0.000 69	10.50
47	ž	0.000 54	10.84
48	ay̆	0.000 38	11.36
49	ă	0.000 36	11.42
50	ɔ̆	0.000 28	11.79
51	ĕ	0.000 14	12.76
52	ʌ̆	0.000 05	14.30
53	aw̆	0.000 05	14.35
54	ɔy̆	0.000 02	15.91

Tab. 3.10: English phonemes, by frequency rank

rank	orthography	phonemic representation	average plog
1	the	ð ə	1.93
2	hand	h ǽ n d	2.15
3	and	ǽ n d	2.20
12640	plumbing	p l ʌ́ m ĭ ŋ	3.71
12641	aerobatics	ɛ́ r ə b ǽ t ĭ k s	3.71
12642	Friday	f r aý d ĭ	3.71
25281	tolls	t ó l z	4.01
25282	recorder	r ĭ k ó r d ɚ̆	4.01
25283	fives	f aý v z	4.01
37922	overburdened	ó v ɚ̆ b ɚ́ d ə n d	4.32
37923	Australians	ɔ̆ s t r eý l y ə n z	4.32
37924	seeps	s iý p s	4.32
50563	retire	r ĭ t aý r	4.75
50564	poorer	p ú r ɚ̆	4.75
50565	vanished	v ǽ n ĭ š t	4.75
63,200	eh	ɛ́	9.07
63,201	Oahu	ó á h ŭ	9.21
63,202	Zhao	ž aẃ	9.25

Tab. 3.11: Examples from English word list, ranked by average plog, bigram model

predicted rank	average reported rank	standard deviation	word
1	6.5	0.96	stations
2	4.17	3.02	hounding
3	4.17	2.97	wasting
4	10.2	5.37	dispensing
5	5.3	3.72	gardens
6	5.3	2.62	fumbling
7	15.5	1.88	telesciences
8	9.8	3.58	disapproves
9	1.8	0.69	tinker
10	12.7	4.35	observant
11	10.1	4.52	outfitted
12	18.7	2.29	diphtheria
13	11	3.27	voyager
14	13.8	4.63	Schafer
15	11.8	3.71	engage
16	16.2	3.71	Louisa
17	19.2	3.76	sauté
18	13.2	5.55	zigzagged
19	12.5	4.64	Gilmour
20	15.7	5.50	aha
21	16.5	4.11	Ely
22	23	0.58	Zhivkov
23	22.2	1.07	kukje

Tab. 3.12: predicted (bigram model) and average reported rank for 23 words of English

rank	phoneme	plog	rank	phoneme	plog
1	#	2.88	21	f	6.03
2	r	3.60	22	ø	6.12
3	a	3.84	23	g	6.18
4	e	3.86	24	v	6.23
5	i	4.03	25	ʊ	6.49
6	t	4.24	26	z	6.50
7	s	4.34	27	u	6.56
8	l	4.52	28	ɔ	6.59
9	k	4.62	29	ʒ	6.69
10	o	4.80	30	ʃ	7.09
11	p	5.06	31	w	7.48
12	m	5.18	32	ɔ	8.48
13	d	5.31	33	œ	8.93
14	n	5.34	34	star	9.14
15	ɛ	5.41	35	!	9.41
16	æ	5.47	36	E	9.96
17	j	5.59	37	U with "	12.13
18	b	5.68			
19	y	5.77			
20	ə	6.01			

Tab. 3.13: French phonemes, by frequency rank

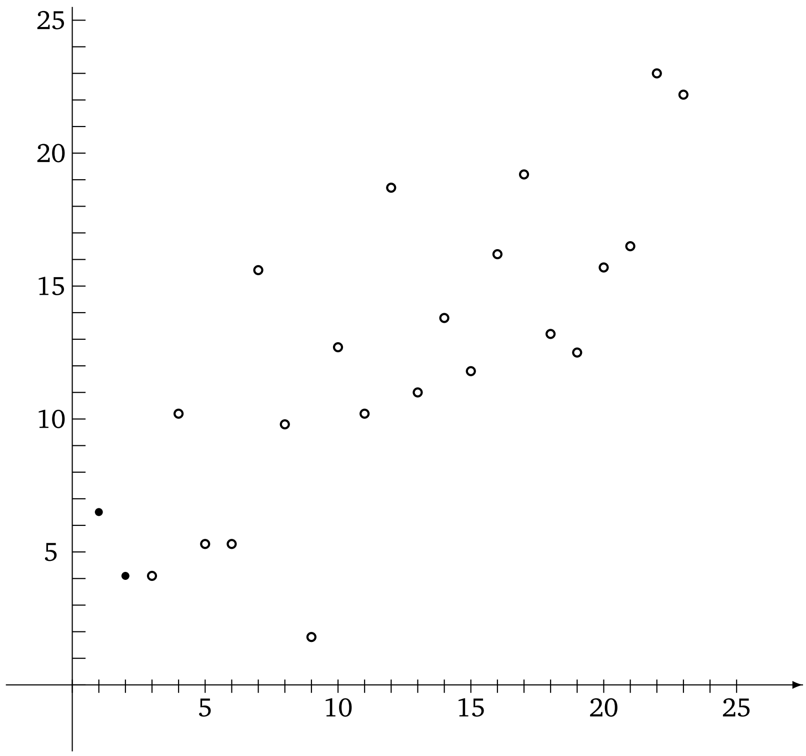
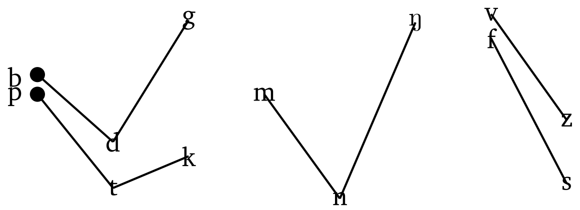


Fig. 3.8: Average reported rank of words in Figure 2

FRENCH NASAL VOWELS

Height	Vowel	example	Vowel	example	Vowel	example
	Front unrounded		Front rounded		Back	
Mid: lax	ɛ̃	plein	œ̃	brun*	ɔ̃	bon
Low:					ã	dans

	labial	alveolar	alveo-palatal	palatal	velar	uvular	laryngeal
Voiceless stop	p	t			k		
Voiced stop	b	d			g		
Voiceless fricative	f	s	ʃ				
Voiced fricative	v	z	ʒ			ʁ	
Nasal	m	n		ɲ	ŋ		
Liquid		l					
Glide	w			j ɥ			



Tab. 3.14: French phonemes

3.12.1 Language identification

If we know that a sample S was produced by one of the languages that constitute our sample of languages, then we want to know what the probability is that language l_i generated it.

Why? Because our goal is to understand the world by finding the model that maximizes the probability of the perceived world. We want a method that will provide us with the way(s) that maximize the probability of the observations. (It's less important to actually compute the probability of the observations; what matters is comparing the models and the probabilities of the data generated by the models. Sometimes you can more easily calculate the relationship between the values $f(x)$ and $g(x)$ (e.g., $\frac{f(x)}{g(x)}$) than it is to compute either $f(x)$ or $g(x)$.)

If we choose a model, we can calculate $pr_{l_i}(S) = p(S|l_i)$. But the probability that language l_i generated S is $p(S|l_i)$, which by Bayes' rule is $\frac{p(S|l_i)p(l_i)}{p(S)}$.

3.12.2 Cross-entropy

We have a corpus C of length N whose letter frequencies we know (because we can count them); the frequency of a typical letter l is $\frac{[l]}{N}$. We can consider various probability distributions π_i over the alphabet Σ. The probability of the corpus, using a unigram probability model and distribution π, is

$$\prod_{l \in \Sigma} (\pi(l))^{[l]}.$$

The logarithm of this quantity is

$$\sum_{l \in \Sigma} [l] \; log \; \pi(l),$$

and the -1 times the average of this quantity is called the entropy:

$$-\sum_{l \in \Sigma} \frac{[l]}{N} \; log \; \pi(l) = -\sum_{l \in \Sigma} freq(l) \; log \; \pi(l)$$

We can visualize the entropy as the inner product of two vectors in $R^{|V|}$. One of the vectors describes a frequency, and hence is on the simplex consisting of points with non-negative coordinates, whose coordinates sum to 1; the other vector is a surface consisting of the points whose coordinates describe the -1 times the logarithms of a distribution (i.e., the surface of all points p such that $\sum 2^{-p_i} = 1$). Each such point describes a probability distribution for our alphabet.

We can vary these two points independently. If we vary the first but keep the second fixed, we may be looking at the entropy of different corpora, assuming the same distribution. If we vary the second but keep the first fixed, we may be looking at the entropy of a given corpus under different assumptions of the distribution that generated it. In that case, we call the quantity that we have calculated the *cross-entropy*. Explain.

3.13 Categories: V, C etc.; HMMs

Suppose we want to divide the symbols of a language into two sets. A large part of what we do when we analyze languages starts that way: we have a large set of elements of some sort; they all have their distinctive characteristics, but let's imagine that we divide them into a small number of groups whose behavior is similar in some particular way.

Our first example of this involves dividing our symbols (letters or phonemes) into two sets. There are many ways to do this. The natural way to understand this goal is to say, what is the simplest model we can think of that assigns a probability to Σ^* and which uses a partitioning of Σ into two sets in order to assign a higher probability to observed sets of data?

Already that critically important word “simplest” has crept in! What do we mean when we say one model is simpler than another? Let’s assume for now that we calculate the complexity of a model by the number of parameters that are used by the model. For example, a unigram with V symbols in its alphabet Σ has $V + 1$ parameters. One of them is the alphabet-size parameter (whose value is V!), and the others are $p(l_i)$, for each letter l in Σ.

We will explore the methods for automatically learning this kind of structure, in part because the natural tool to use first is a *hidden Markov model*, or HMM, and they are very useful for many problems in machine learning: if you have a problem that can be solved with an HMM, then you are in fine shape; and if you have a problem that cannot be solved with an HMM, then it is helpful to understand exactly why an HMM is not powerful enough.

Our main reading on this is from the Manning and Schütze book. Here I will add some supplemental material.

3.13.0.1 Hidden Markov models: finding classes of letters

Brief overview of HMMs, to which we will return in more detail. HMMs present us with the first case in which we talk about non-integral counts (which we can also call *expected counts*). This involves the case where we understand observed data (which normally we would count with integral counts) as containing only a partial specification of the “reality” we’re interested in: reality contains further parameters whose values we can’t directly observe. So we use a model to make statements about how often we expect the system to be in various states, given the observations that we in fact make.

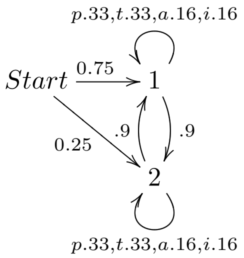
Here is a non-hidden markov model: given the outputs, you know the path it takes through the graph.

$$\begin{tikzcd}
Start \arrow[r, "a"] \arrow[dr, "b"] & A \arrow[r, "a"] \arrow[dr, "a"] & A \arrow[r, "a"] \arrow[dr, "a"] & A \\
& B \arrow[r, "b"] \arrow[ur, "b"] & B \arrow[r, "b"] \arrow[ur, "b"] & B
\end{tikzcd}$$

Here you don’t:

$$\begin{tikzcd}
Start \arrow[r, "0.75"] \arrow[dr, "0.25"] & 1 \arrow[r, "aipt"] \arrow[dr, "aipt"] & 1 \arrow[r, "aipt"] \arrow[dr, "aipt"] & 1 \\
& 2 \arrow[r, "aipt"] \arrow[ur, "aipt"] & 2 \arrow[r, "aipt"] \arrow[ur, "aipt"] & 2
\end{tikzcd}$$

Let us suppose that in State 1, the probability of generating *p is 1/3, t 1/3, a 1/6, i 1/6,* and in State 2, the probability of generating p is *1/6, t 1/6, a 1/3,* and *i 1/3*.



Then we want to be able to answer questions like (and we *can* answer them): What is the probability of emitting the string #p (i.e., the string starting 'p')? The answer is: it's the sum of going through the paths Start-1-2 and emiting p, Start-1-1 and emitting p, Start-2-1 and emitting p, and Start-2-2 and emiting p. Which is: $.75(1/3) + .25(1/6) = 7/24 \approx 0.29$. (You can see I summed together the probabilities of the paths State-1-1 and Start-1-2, that is, .75(1/3)(0.1) + .75(1/3)(0.9)). Or we can ask: what is the probability of being in state 1 after emiting #p? The answer to that is .75(1/3)(0.1) + 0.25(1/6)(0.9) = .025 + .0375 = 0.0625.

And now we can turn that into soft counts. That is, if we know the probability of being in State 2 after emiting #p just like we know the probability of being in State 1 after emiting #p, then we distribute the count of 1 over those two paths, in proportion to those probabilities.

Prob(being in state 2 after emiting #p) = 0.25(1/6)(.1) + 3/4(1/3)(.9) = 0.225 + 0.004167 = 0.229.

So the sum of the probabilities of being in states 1,2 after emiting #p is 0.0625 + 0.229 = 0.2915.

So now we can assign softcounts. The softcount of generating #p and being in State 1 is $\frac{.0625}{.2915} = 0.2144$, while the softcount of generating #p and being in State 2 is $\frac{.229}{.2915} = 0.7855$.

3.13.1 Syllables

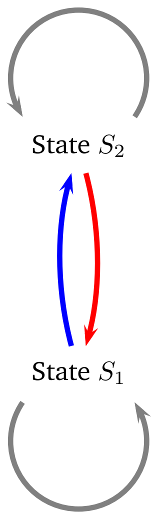
3.13.2 Vowel harmony

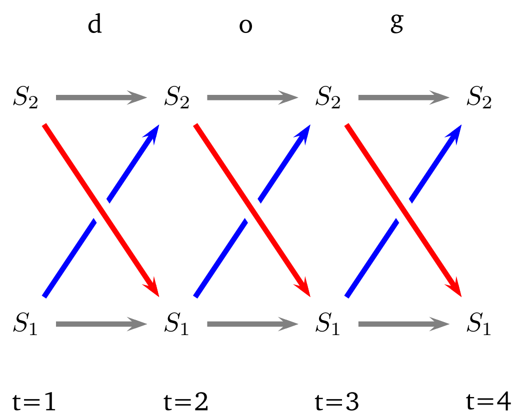
3.14 Hidden Markov Models (HMMs)

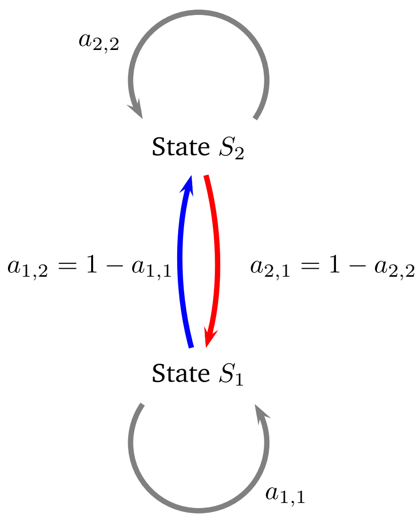
3.15 The problem

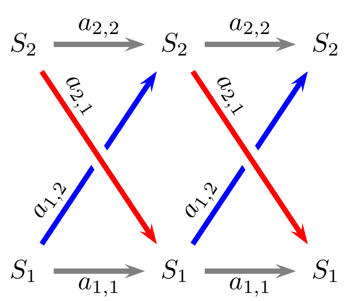
$\mathbf{X}$ = sequence of random variables (X_i).	There are N states: $S = S_1 \ldots S_N$. N=2 in these diagrams. The random variables taken on the states as their values.
O = $\{o_i\}_{i=1,T}$	Output sequence (letters, e.g.).
T	Number of symbols output—so we care about T+1 states.
Π	Initial probability distribution over the states.
A	Transition probabilities from state to state.
B	Emission probabilities: $b_{x_i o_i}$.

o_i is selected from our alphabet $\mathcal{A}$. For our project, the alphabet is letters, but you could build an HMM where the "alphabet" was words, i.e., the lexicon (vocabulary) of the language.









$$\text{Markov model on states: limited lookback (horizon)} : p(X_{t+1} = s_i | X_1 \dots X_t) = p(X_{t+1} = s_i | X_t) \tag{3.20}$$
$$\text{Stationary} p(X_{t+1} = s_i | X_t) = p(X_2 = s_j | X_1) \tag{3.21}$$
$$\text{Transition matrix:} a_{ij} = \tag{3.22}$$

So for fixed i,

$$\sum_{j=1}^{|S|} a_{ij} =$$

We initialize

$$p(X_1) = \pi_i.$$

So (what are we summing over?)

$$\sum \pi_i =$$

3.16 Compression and encoding

3.16.1 First notions of optimal encoding

> If we know (or think we know) the structure of the device of the device that produced the message M, we can compress M as Ann *sends* it to Betty.

Claude Shannon developed a set of ideas known as *information theory* which had their first concrete application in the transmission of messages by digital means. The basic idea concerned a situation in which Ann wants to sends Betty a message from a language: let's say the language is English, and as a first approximation, we will assume that the message is a concatenationation of words from a prespecified lexicon (word-list) called $\mathcal{L}$. Technology requires that the message be just a series of 0's and 1's—that is, a string from $\{0,1\}^+$. Ann therefore needs an **encoding**, i.e., a map from $\mathcal{L}$ to $\{0,1\}^+$, and the map must be **injective** (no two words are encoding by the same binary string) but not necessarily surjective. In fact, it will *not* be surjective, because there is a natural condition that both Ann and Betty agree is necessary: the encoding must be **prefix-free**: we place the constraint that the encoding of word w can never be the prefix of another word v (i.e., if w encodes as 0101, then no other word can encode as 01011). In this way, the encoding is **instantaneous**: after any digit, Betty knows if she has gotten to the end of the encoding of an individual word.[26]

[26]For now, that conclusion comes immediately from looking at the sequence of bits Ann has just sent. When we get to arithmetic encoding, we will see that Betty has to do some computation to reach that conclusion, but she does not need to look ahead to future bits of the message.

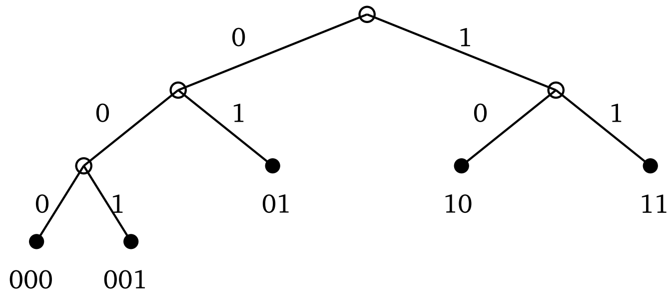


Fig. 3.9: binary tree of encodings with prefix property

Prefix-free encoding systems can be organized into a binary tree in which all encodings correspond to terminal nodes of the tree, where the string indicates the path taken from the root to the leaf: a 0 means take a left branch, and a 1 means take a right branch. If any non-terminal node were used as an encoding and it dominated a terminal node that were used as an encoding, then clearly the system would not be prefix-free. When we are talking about a particular encoding, we will talk about the nodes of the tree that represents it, identifying strings and nodes in the natural way. You should feel comfortable with the statement that there is an equivalence between a description of a set of strings which are jointly prefix-free (on the one hand) and the description of a set of strings as the edge-labels of paths going from the root of a tree all the way to its leaves.

3.16.2 Addition April 2018

Let's think about encoding from a base-10 point of view (with 10 digits and base 10 logs), and then transfer the ideas to the binary world, where nothing essential is different at all.

We know that the cross-entropy is always greater than the self-entropy, and here that means that if we have an encoding system which uses $-logp(w)$ bits to encode w, we have an optimal system, among those that have the prefix property (= are prefix-free).

We have a set of probabilities for the messages of our systems, and without real loss of generalization, we can say we have a set of probabilities for the letters of our alphabet, and an ordering for the symbols too (i.e., we know how to alphabetize).

Then any string can be associated with an interval (p, q), where $q - p$ is the probability of S. Our goal is to devise a system that allows us an optimal encoding as a string of decimal digits (decimal for now; after that, binary). This means proposing a map between strings of digits and strings in the alphabet

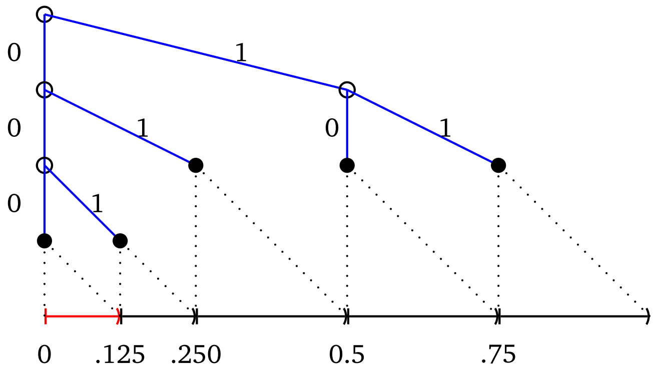


Fig. 3.10: [0.0,0.001)= [zero, one eighth) = [0,.001) = from_string_to_interval(000)

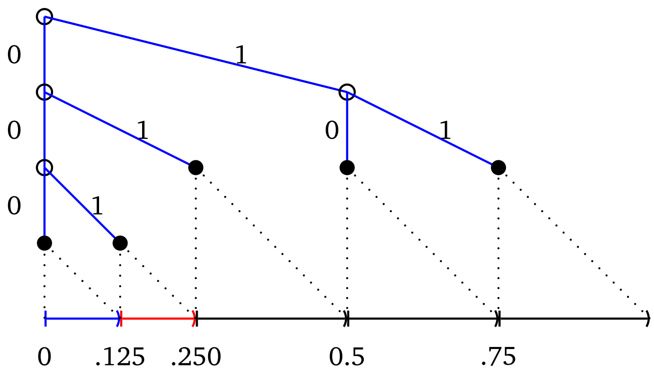


Fig. 3.11: [0.125, .250)= [0.001,0.10] = from_string_to_interval(001)

Example 1. If there are 10 letters ($a - i$ plus #)and a uniform probability over the symbols, then of course the encoding is simple: *cab#* has probability $10^{-}4$, i.e., .000 1, which corresponds to the string *0001*.

Example 2: Same letters but probabilities are not uniform. We compute a probability of 0.000 05 for string cab#, and it corresponds to an interval [0.010 j

3.16.3 Return to older text

If we have a message, like Paul Revere, which is one of just two possible messages (land or sea) which have equal prior probabilities, then we are in a situation like this:

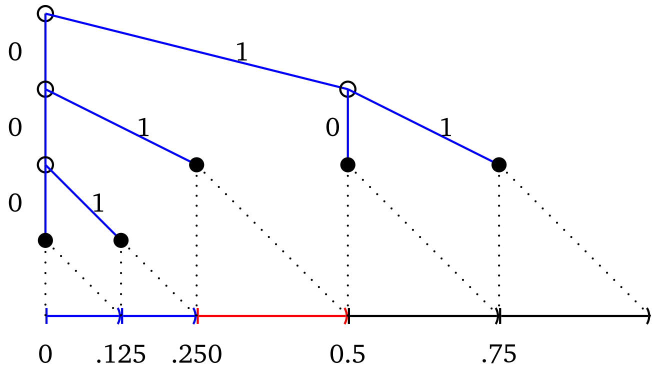


Fig. 3.12: [0.25, 0.5)= [0.01,0.1] = from_string_to_interval(01)

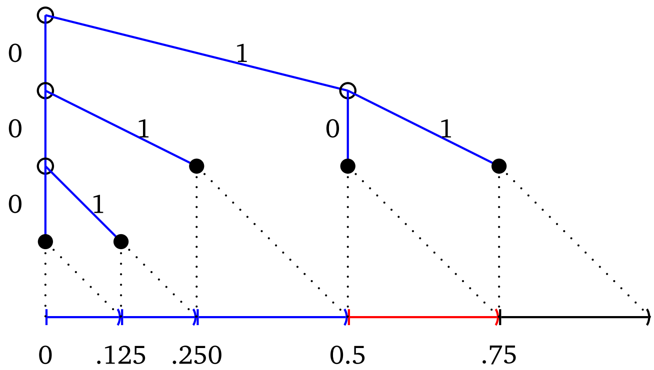


Fig. 3.13: [0.5, 0.75) = [0.1,0.11] = from_string_to_interval(10)

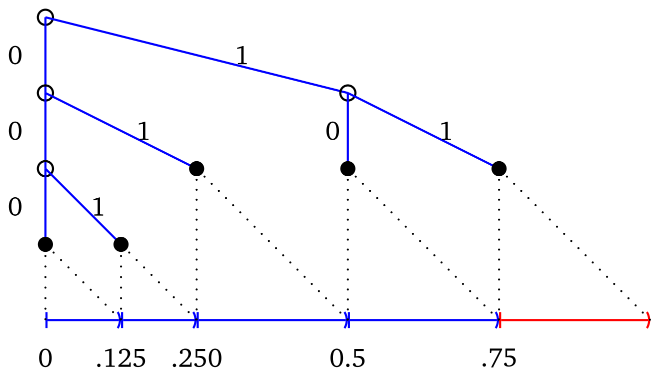


Fig. 3.14: [0.75, 1.0) = [.11,1.0) = from_string_to_interval(11)

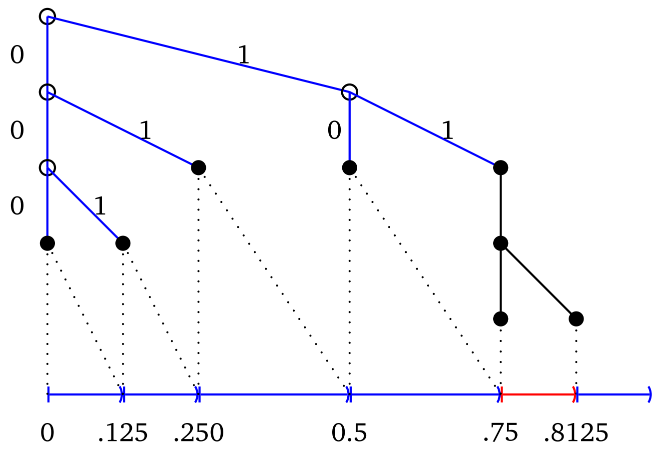


Fig. 3.15: [0.75, 0.8125) = (three quarters, thirteen sixteenths) = [.11,.1101)= from_string_to_interval(1100)

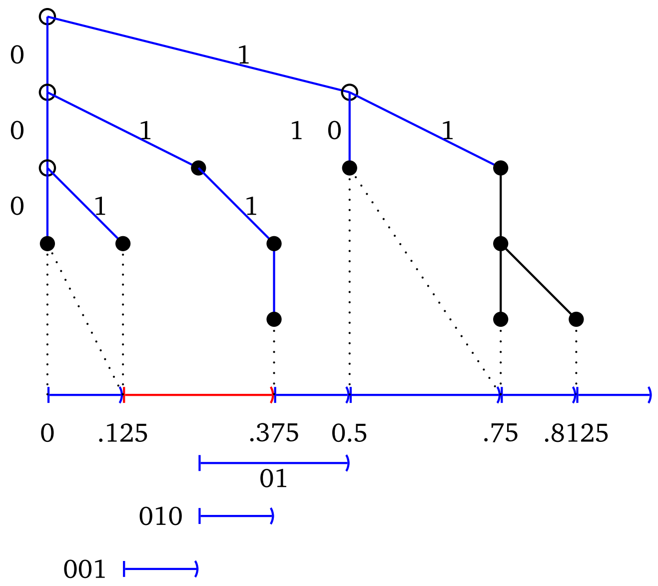


Fig. 3.16: [0.125, .375) = (one eighth, three eights) = [.001,.011); width = 0.01

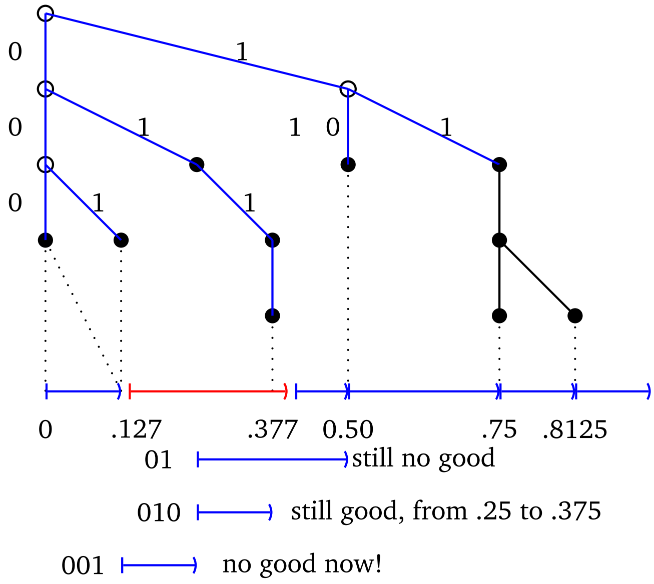
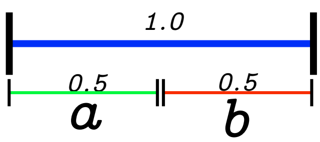


Fig. 3.17: [0.127, .377) = (one eighth, three eights) = [.001,.011); width = 0.01

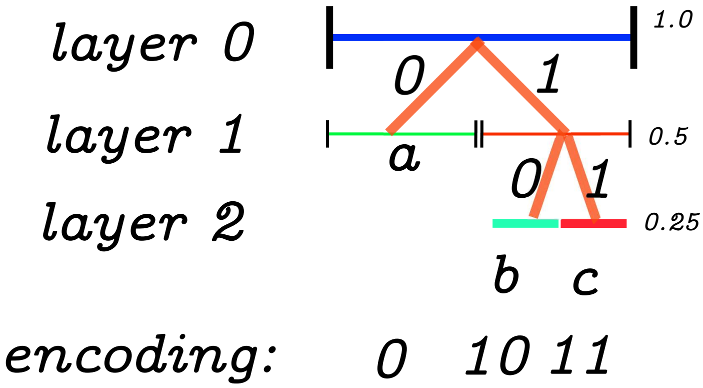
Best encoding?

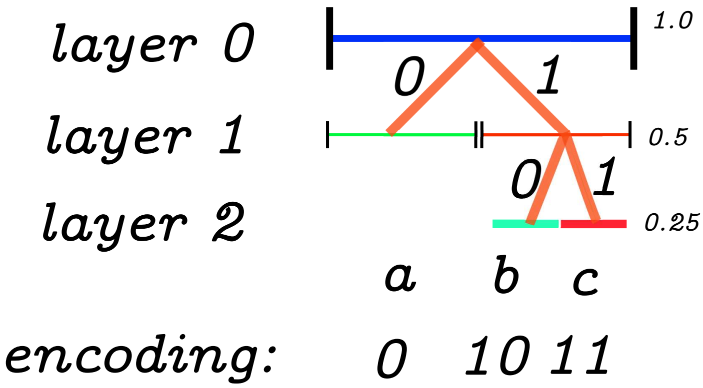
and the best encoding is to use just 1 bit: either 0 or 1, depending on which message you want to communicate.

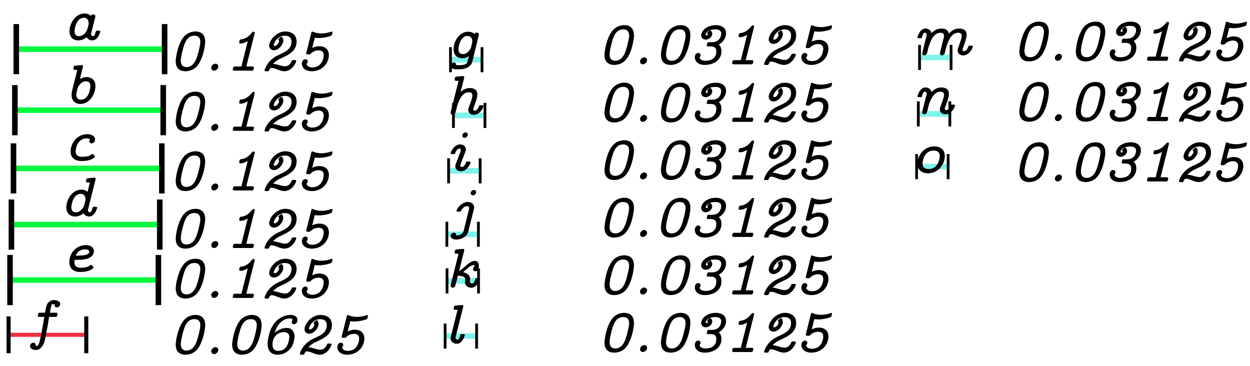


In this simple case, each of the two possible messages has a probability which is an integral power of 2: each of them has probability 2^{-1}. We are going to consider several cases now in which there are more than two possible words we which to communicate, but every one of them has a probability which is an integral power of 2 (a

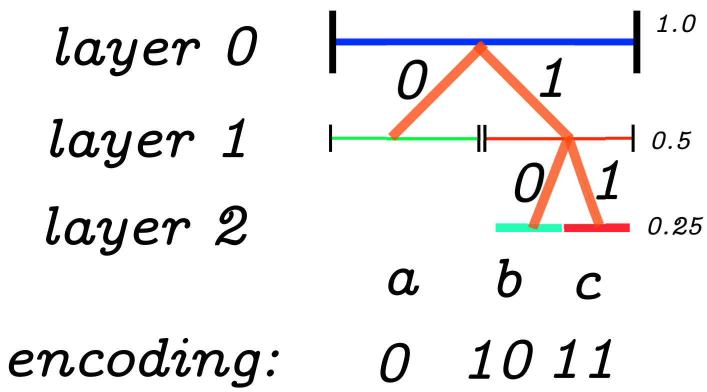
We can imagine a red binary branching tree, with a 0 associated with each left-hand branch and a 1 associated with each right-hand branch. Then any path through that tree will be described as a sequence of 0s and 1s, and it will end on one of the leaves of the tree; that sequence will be the encoding for the word on the leaf.



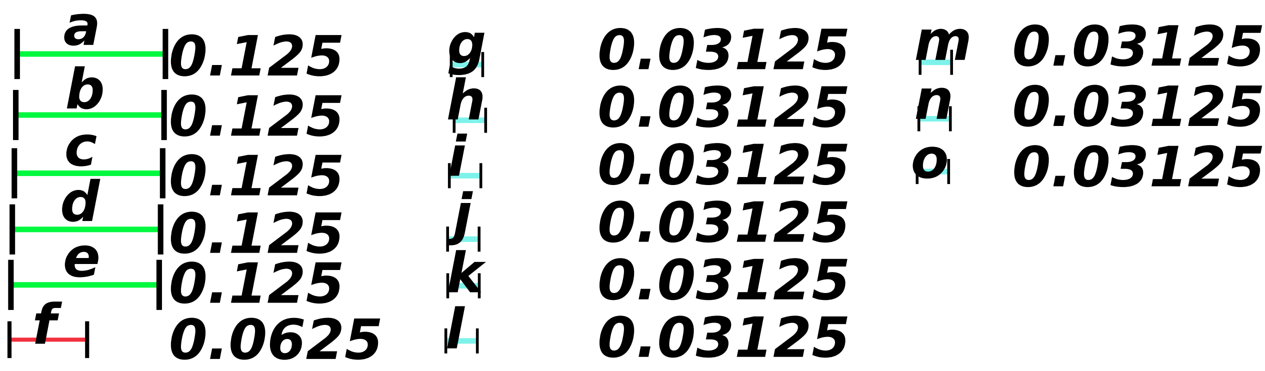


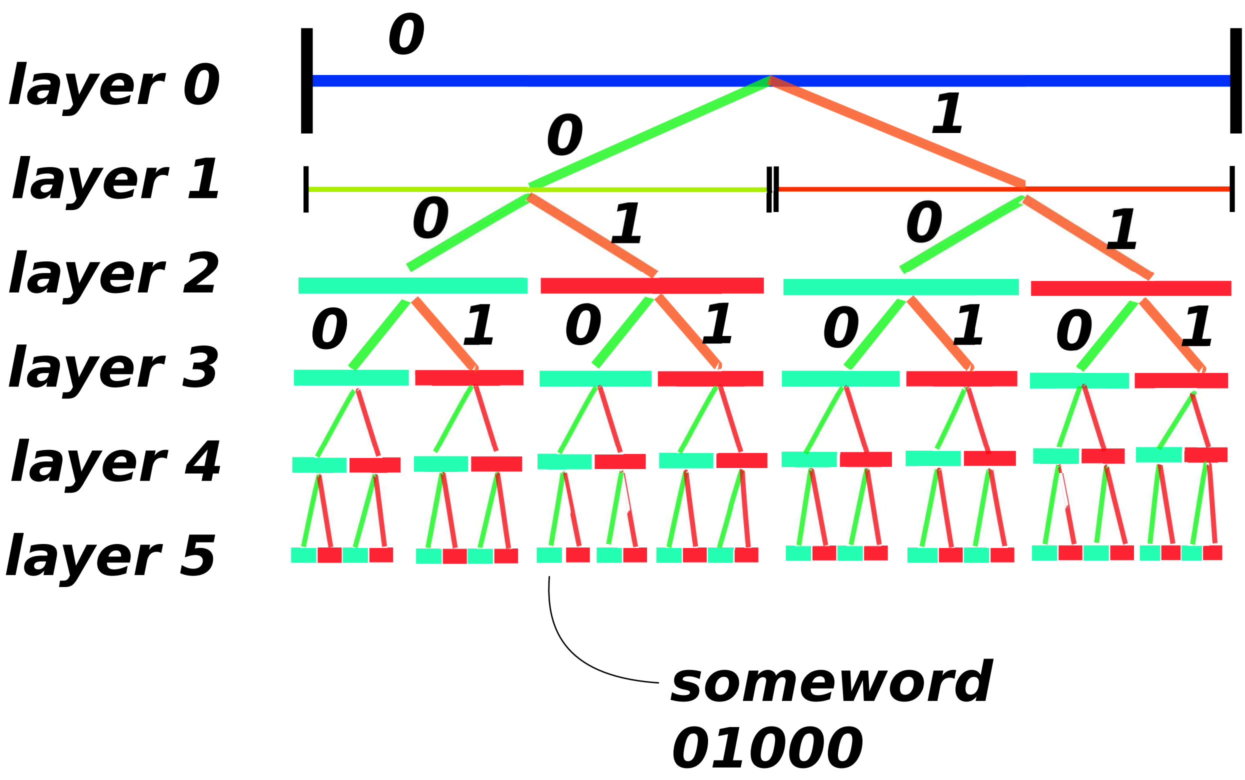


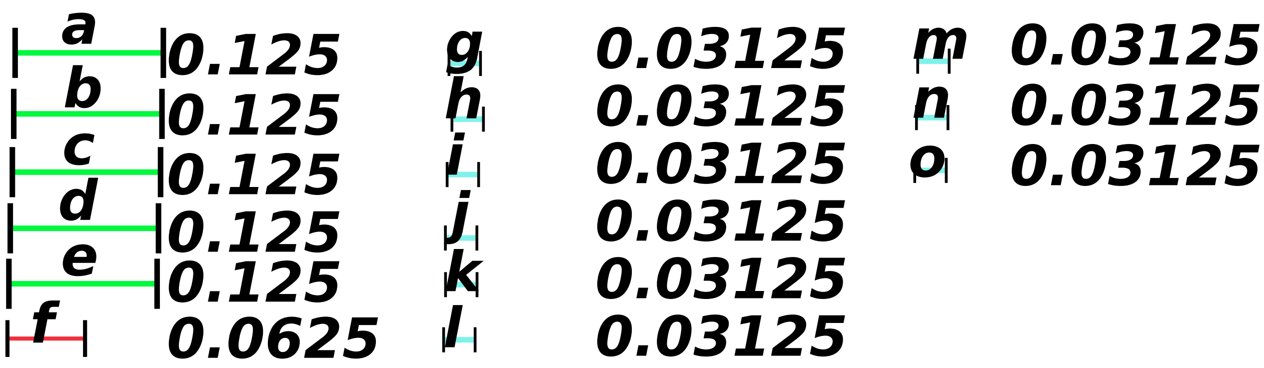
a 0.125
b 0.125
c 0.125
d 0.125
e 0.125
f 0.0625
g 0.03125
h 0.03125
i 0.03125
j 0.03125
k 0.03125
l 0.03125
m 0.03125
n 0.03125
o 0.03125

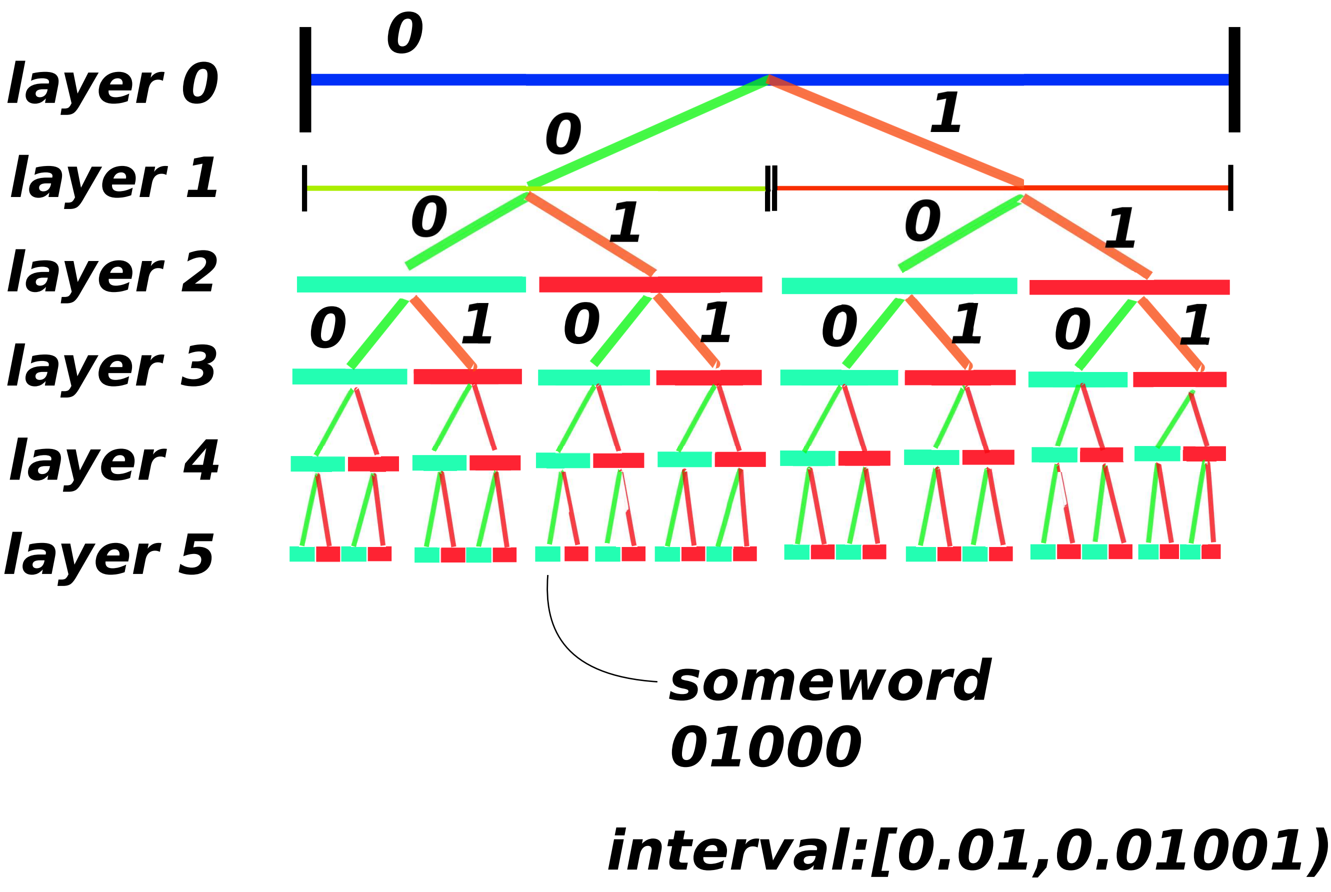


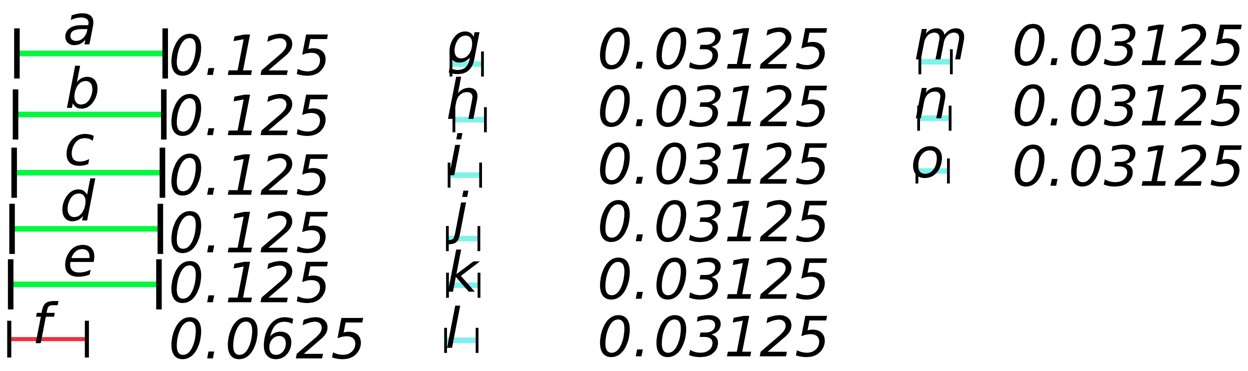
layer 0
1.0
0
1
layer 1
0.5
0
1
layer 2
0.25
a
b
c
encoding:
0
10
11

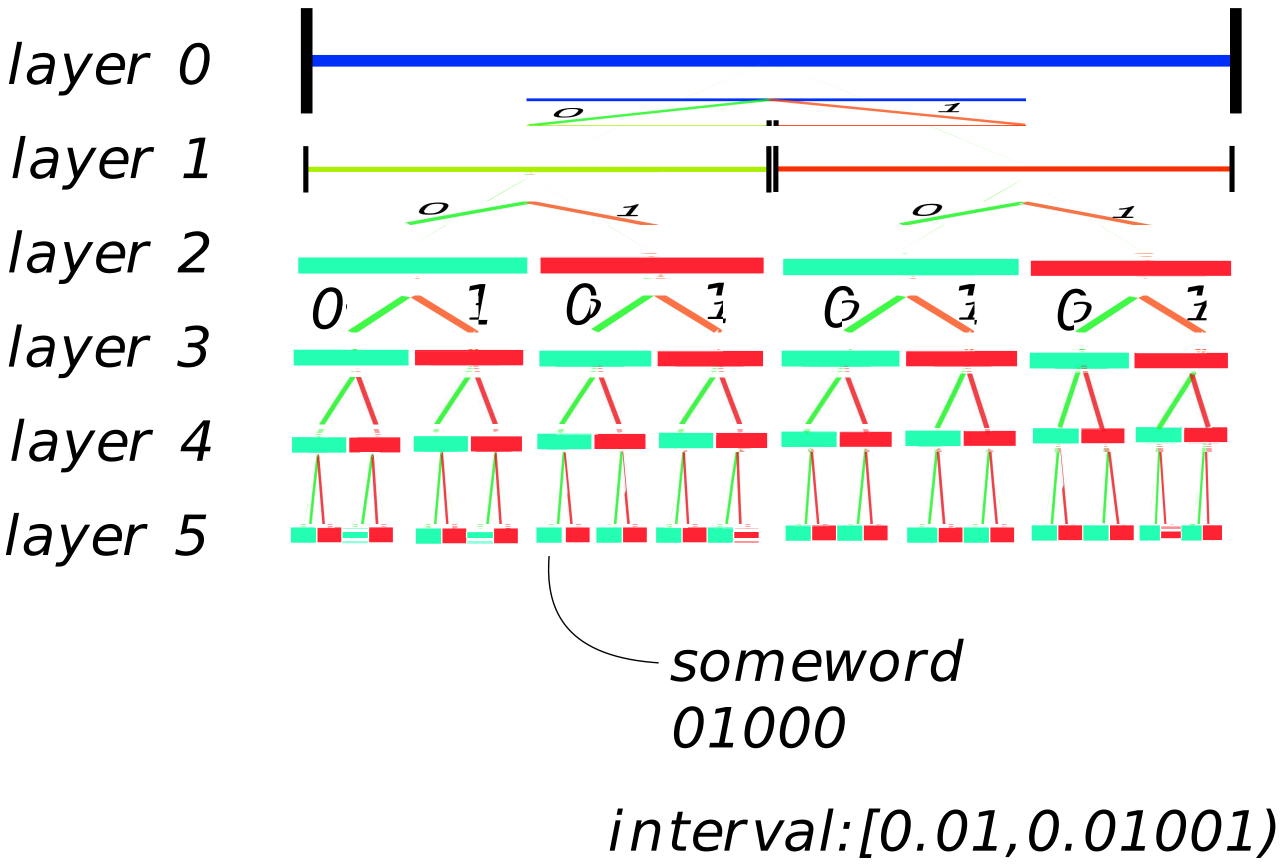

a 0.125
b 0.125
c 0.125
d 0.125
e 0.125
f 0.0625
g 0.03125
h 0.03125
i 0.03125
j 0.03125
k 0.03125
l 0.03125
m 0.03125
n 0.03125
o 0.03125

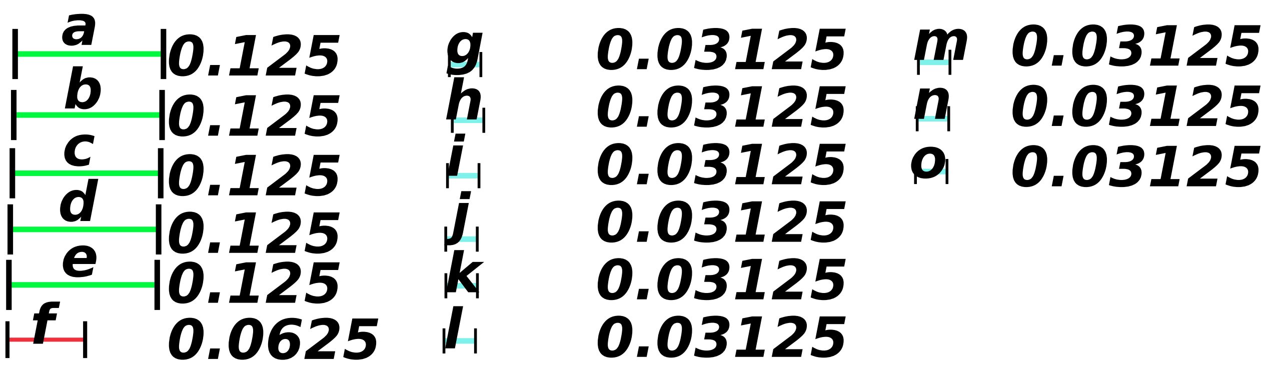

layer 0
layer 1
layer 2
layer 3
layer 4
layer 5
0
1
someword
01000

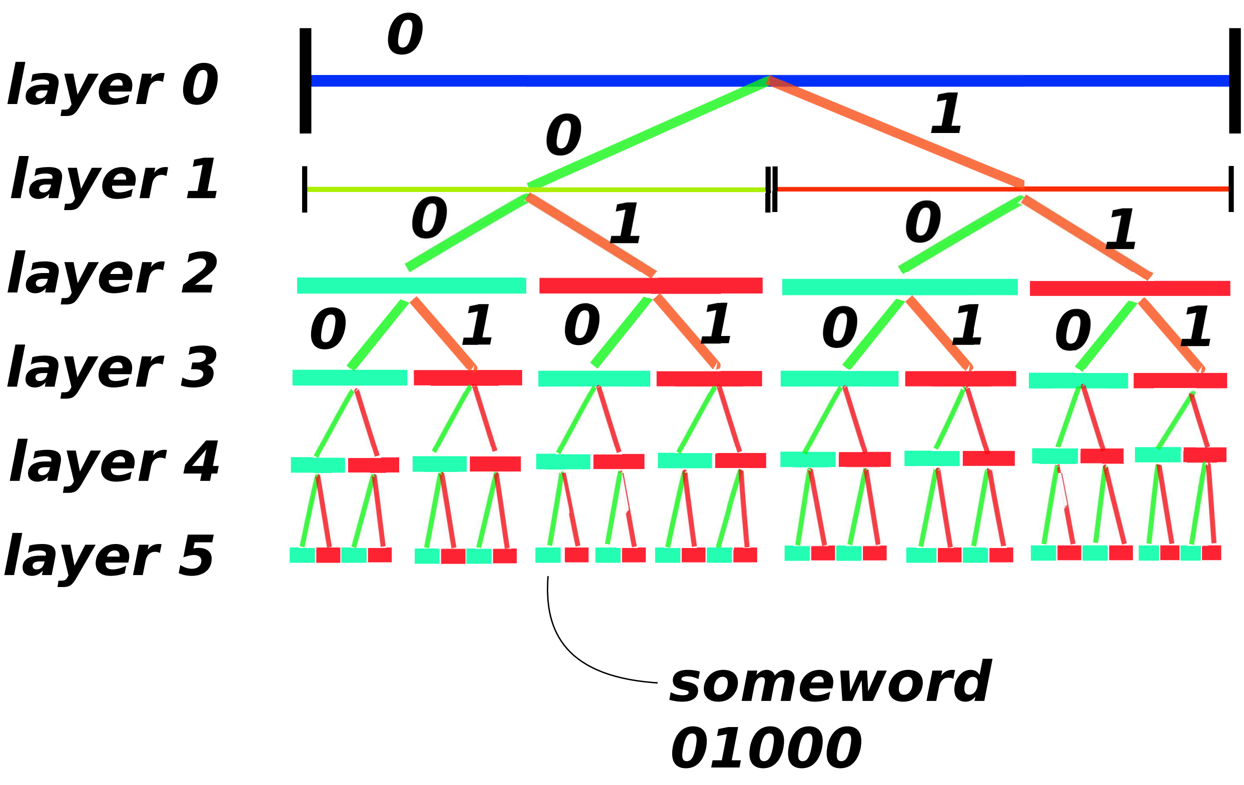

a 0.125
b 0.125
c 0.125
d 0.125
e 0.125
f 0.0625
g 0.03125
h 0.03125
i 0.03125
j 0.03125
k 0.03125
l 0.03125
m 0.03125
n 0.03125
o 0.03125


layer 0
layer 1
layer 2
layer 3
layer 4
layer 5
0
1
someword
01000
interval:[0.01,0.01001)


a 0.125
b 0.125
c 0.125
d 0.125
e 0.125
f 0.0625
g 0.03125
h 0.03125
i 0.03125
j 0.03125
k 0.03125
l 0.03125
m 0.03125
n 0.03125
o 0.03125

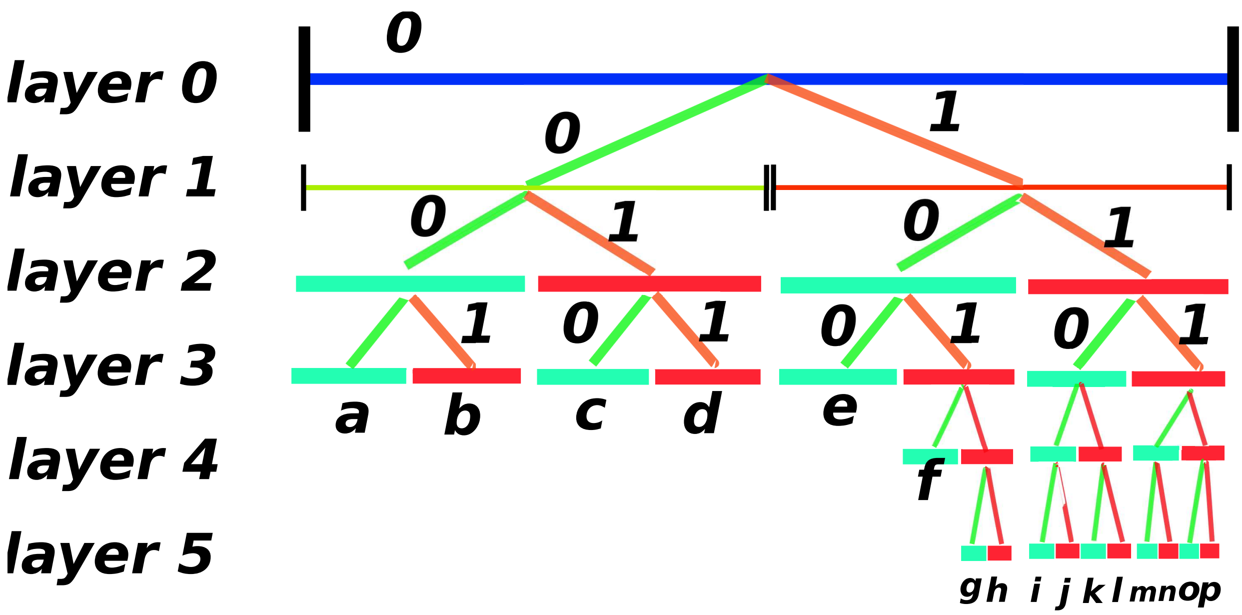

layer 0
layer 1
layer 2
layer 3
layer 4
layer 5
0
1
someword
01000
interval:[0.01,0.01001)


a 0.125
b 0.125
c 0.125
d 0.125
e 0.125
f 0.0625
g 0.03125
h 0.03125
i 0.03125
j 0.03125
k 0.03125
l 0.03125
m 0.03125
n 0.03125
o 0.03125


layer 0
layer 1
layer 2
layer 3
layer 4
layer 5
0
1
someword
01000

When	Uniform prob	p(x[i])	p(x[i] | x[i-1])
Shannon era	0 order Markov	1st order Markov	2nd order Markov
Later (now)		0 order Markov	1st order Markov
Today	uniform	unigram	bigram

Tab. 3.15: Terminology: What order Markov model?



We are interested in properties of good encoding systems, and one of the important qualities of an encoding system is its ability to create relatively short strings (from $\{0,1\}^+$, *given* a particular message from $\mathcal{L}^*$). If we knew nothing about the frequencies of the different words in $\mathcal{L}$, we could easily create an encoding scheme which would limit the worst case length, by encoding each word by a binary string of length $\lceil log_2|\mathcal{L}|\rceil$ (where $|\mathcal{L}|$ is the number of items in $\mathcal{L}$). We can always enumerate the $|\mathcal{L}|$ different members of the lexicon and we will need no more than the base 2 log of their count to do so, rounding up as necessary.

In discussing communication systems of this sort, we typically assume, however, that we know certain statistical properties of the messages that Ann sends, such as the (time-averaged) frequency of his usage each individual word in $\mathcal{L}$, $fr(w_i)$. In that case, we can come up with much better encoding systems.

Consider again the binary tree of the encodings of a prefix-free encoding systems. If all nodes are either terminal or binary branching, we say that the tree is *complete* (every binary string either identifies a node in the tree, or has a prefix which identifies a terminal node in the tree). There is a natural way to associate each node in our tree with a subinterval of [0,1): if the string is s (e.g., 01010), then we associate it with the interval that begins with the binary fraction we would write

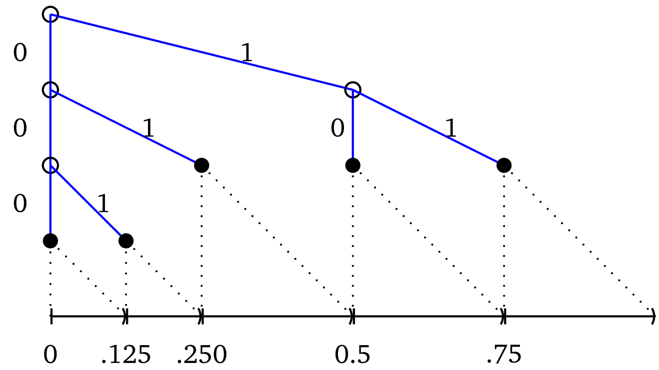


Fig. 3.18: The canonical correspondence

as "0.s" (call that number p), of length $2^{-|s|}$; so the interval is $[p, p + 2^{-|s|})$. This is very natural, given the figure; see the figure on the canonical correspondence.[27]

Now, it is not hard to show that if we have a lexicon $\mathcal{L}$ whose members w_i all have frequencies which are (negative) powers of 2, then the very best encoding system that can be devised is one satisfying the simple property that the length of the encoding of word w_i is $-log_2 fr(w_i)$. Why?

Recall that a distribution π over a set S assigns a non-negative member of [0,1] to each member such that $\sum_x \pi(x) = 1$. Let's use the term "plog distribution $\pi()$" to mean the function $-log_2\pi()$ ("plog" standards for "positive log"). Plog distributions turn out to be central to information theory. You can see that a plog distribution maps from the members of the set to the positive reals. If we have a (complete, prefix-free) encoding $\mathcal{E}$, then the canonical correspondence defines a distribution, and *its* plog distribution is the lengths of each element of the encoding (i.e., the lengths of the strings corresponding to the paths through the tree).

The cross-entropy H(P,Q) between two distributions P and Q over the same set X is defined as $-\sum_{x \in X} P(x) log_2 Q(x)$: you can see that this is the sum of the products of the corresponding values of the distribution P and the values of Q's *plog distribution*. And it turns out that this quantity is always strictly larger than H(P,P) for any distribution $Q \neq P$:

$$\sum_x p(x) ln \frac{q(x)}{p(x)} \leq \sum_x p(x) \left(\frac{q(x)}{p(x)} - 1 \right) \tag{3.23}$$

28

[27] See Li and Vitányi, *Introduction to Kolmogorov Complexity*, the standard book in this area.
[28] Why? Look at the plot of $ln(x)$, and compute its first and second derivatives, and its value at (1,0): $ln(x) \leq x - 1$.

$$= \sum_x p(x)\frac{q(x)}{p(x)} - \sum_x p(x) = 1 - 1 = 0. \quad (3.24)$$

So $\sum_x p(x)ln(\frac{q(x)}{p(x)} \leq 0$, and the same holds then if we change the base of the logarithm to 2. Therefore $\sum_x p(x)log_2 q(x) \leq \sum_x p(x)log_2 p(x)$, and multiplying both sides by -1, we see that $H(P,Q) \geq H(P,P)$.

Terminology: the quantity H(P,P) is known as the *self-entropy* of P, or simply as the *entropy* of P.

So what we have seen is this: any encoding system E maps to a distribution, and in particular to a *plog distribution* whose values are the lengths of the encodings. If we encode a message M of length $|M|$using E, then the expected number of bits of the encoded message will be $|M| \sum_{x \in \mathcal{L}} p(x)(-log\, q(x))$. Better put, the average encoding length of a message drawn by distribution P but encoded in a system derived from distribution Q in the way we have just sketched is the cross-entropy $H(P,Q)$. And we have already proven that the cross-entropy can never be shorter than the self-entropy (and will in fact be larger, unless P=Q).

We can now speak of the optimal compressed length of data d given a model (grammar) h that generates d and assigns a probability $p(d)$: it is $-log_2 pr_h(d)$. We'll express that as $|d|_h$. It is a length, whose unit of measure is the bit.

3.16.4 Comparing an HMM to a probabilistic finite-state automatmon (FSA)

Suppose we are interested in some words of length 4; we will start with "bill" and "trip". Let us imagine an FSA with four states. One of them is the start state; only one of them is an accepting state. There is a very particular linear order to the graph of this FSA: an edge from state 1 to state 2, from state 2 to state 3, from state 3 to state 4, and no other edges. Each node is associated with an emission probability, and the transition probabilities are trivial. Let's suppose that initially the emission probabilities form a uniform distribution over the 26 letters of the alphabet for all states. Then the probability of each four letter word initially is $\frac{1}{26^4}$, and we get that by multiplying all the a's and b's just as before, only the a's are all 1.0, and the π is trivial too.

Suppose we now calculate the counts (like soft counts, but they are hard, not soft, now! nothing is hidden), just for the HMM. That means we will add up the soft counts on a big SC table. Each state will have associated with it a single row, and counts of all the letters that it emitted:

state	emission	count
1	b	1
1	t	1
2	i	1
2	r	1
3	l	1
3	i	1
4	l	1
4	p	1

Now we can recompute (this is *maximization*!) the emission probabilities of each state. Here is what we get for the first three states (you do the fourth):

state	emission	prob	state	emission	prob	state	emission	prob
1	a	.0	2	a	.0	3	a	.0
1	b	.5	2	...	.0	3	...	.0
1	...	.0	2	i	.5	3	i	.5
1	t	0.5	2	...	0.5	3	...	.0
1	...	0	2	r	0.5	3	l	0.5
1	z	0	2	...	.0	3	...	0

What just happened? Essentially this: we have now created an FSA that generates the training data with the largest possible probability. Each word has probability $\frac{1}{2^4}$. Why is that maximal? Why isn't it 0.5 probability for each word?

Let's consider for a moment a different model, in which each state has the *same* probability distribution: we say that the variables are *tied*. We are doing this just for educational reasons, so we understand what happens when we do this. There's nothing inherently interesting about this assumption.

We would build a table in which we would not distinguish between the states, so it would be simply this (compare to the one above):

state	emission	count
?	b	1
?	i	2
?	l	2
?	p	1
?	r	1
?	t	1

And maximization gives us frequencies:

state	emission	count
?	b	.125
?	i	.25
?	l	.25
?	p	.125
?	r	.125
?	t	.125

I put '?' to remind you that we are not distinguishing between states: we have tied their values together. Question: what is the probability assigned to "bill"? And "trip"?

Symbol	Probability	Range
a	0.2	[0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
$	0.1	[0.9, 1.0)

Tab. 3.16: BCW example: arithmetic encoding

3.16.5 Lempel-Ziv: the de facto standard

3.16.6 Arithmetic encoding

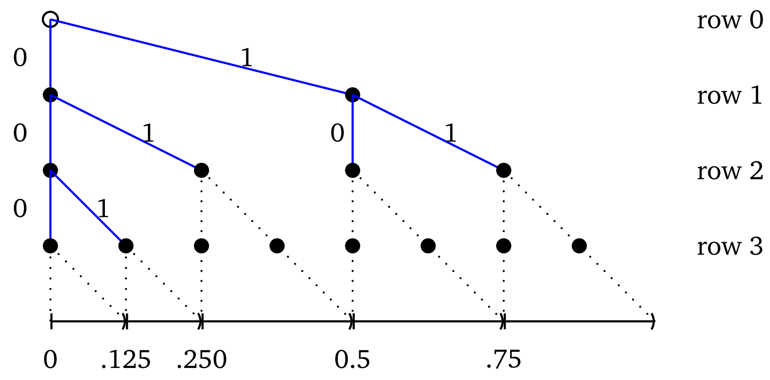
Arithmetic encoding was independently discovered by about 5 different people during the 1970s.[29] It is conceptually elegant and useful in practice too, and clarifies some of the ideas we have touched on so far. Arithmetic coding does not use an encoding in the sense that we have defined it above; it employs an algorithm to map strings in Σ^* to strings in $\{0,1\}^+$.

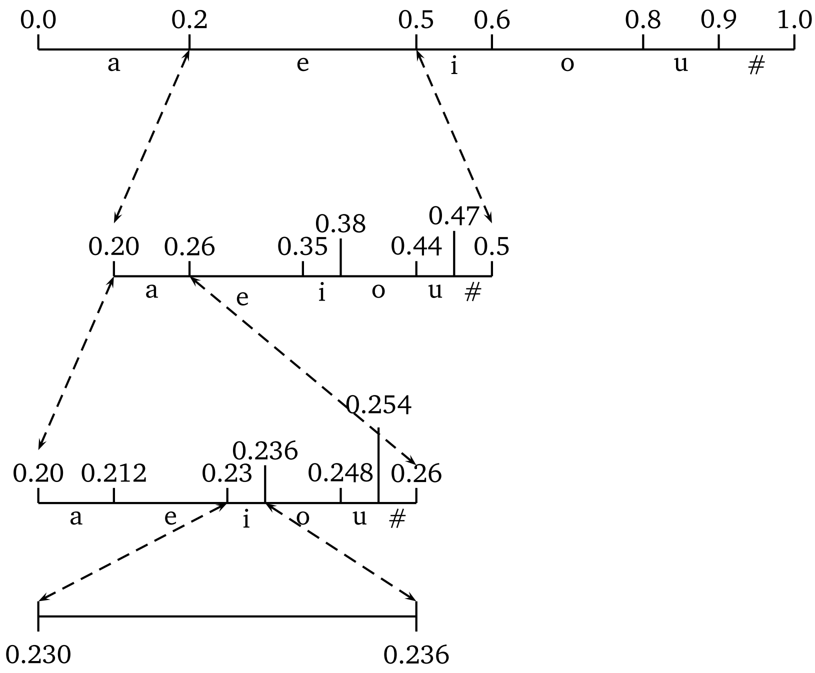
The fundamental insight behind arithmetic encoding is that if the sender and the receiver share a model that assigns a probability to each possible message from a countable set, then we can use the model to associate with each possible message m an interval I_m, and these intervals partition [0,1). To send a message m, then, we need only send the shortest (shortest, not smallest) binary string that corresponds to a real in I_m.

We can do this with pretty much any way of assigning a probability distribution over a countable set. If we want to send message m which is the i^{th}, we sum the probabilities of the messages with lower index numbers: q = $\sum_{j=1}^{i-1} p(j)$, and define m's interval as $[q, q + p(m))$.

That's a bit abstract. It's clearer in a simple case, like a unigram model. Let's suppose we are sending the message *eaii* (i.e., *eaii#*). Then we drill down to smaller and smaller intervals. First, the table tells us that to encode e, we want to restrict our attention to [0.2, 0.5). Now we take that interval, and divide it up with exactly the same proportions as before (since this is a unigram model: if we were using a bigram model, we would divide e's interval up in a way different than how we divide up i's interval). We see that the interval corresponding to ea is [0.20, 0.26). We then divide that smaller interval up, and we see that corresponding to i is the interval [0.23, 0.236), and next the subinterval corresponding to the 2nd i in $eaii$ is [0.233, 0.2336). Finally we break up that interval to find the part corresponding to #, which is [0.23354, 0.2336). Now we just send the shortest binary string which corresponds to an interval entirely inside that calculated interval.

[29]The very best source on text encoding of all sorts is *Text Compression*, by Bell, Cleary, and Witten.





See (i.e., read) the discussion of arithmetic encoding (read the material placed on line excerpted from Bell, Cleary, and Witten, *Text Compression*.)

More discussion of arithmetic encoding; brief example of finding the interval corresponding to the sequence *ab*, given a distribution (2/3, 1/4, 1/12) for a, b, c. [30]

3.16.6.1 Different description, similar material

In what follows, we will use the symbol s to refer to a string in $\{0,1\}^+$ and put a hat on it ($\hat{s}$) to indicate the number represented by this string of binary digits to the right of the binary point.[31]

We want to be able to easily talk about all of the positive "binary-rational" numbers, those of the form $\frac{m}{2^{-k}}$, with m odd. Imagine a grid (as in Figure 3.18) where each such number appears a grid-row: in particular, on the k^{th} grid row. 0 appears by default (so to speak) on all rows. So any binary string s maps to $\hat{s}$, which is a binary-rational, and we will represent the inverse of that hat-function by σ (think *string*): σ maps a binary-rational to a binary string.

We can also unambiguously talk about a binary-rational number's $predecessor$: it is the binary-rational to its left on the same row, which is to say, $\hat{s}$'s predecessor is $\hat{s} - 2^{|s|}$. Let's write it $Left(\hat{s})$.

Here is an important thing to know: if we have an interval I of length $\Delta = 2^{-k}$, call it $[x, \Delta)$, with no restriction on x, then you can find a string of length no greater than k+1 which maps (via the canonical mapping) to an interval entirely inside that interval I.

Here is how we see that. If x is itself a binary-rational number (that is, of the form $\frac{m}{2^{-k}}$) and $m \leq k$ (which is to say, x is one of the points on the grid on the k^{th} row or above) then the string that we are looking for is simply $\sigma(x)$, padded with enough 0's on the right to make up a string of length k.

Suppose that x is not such a coarse number: it might be a binary-rational on a lower row, or simply not a binary rational at all. Clearly there is a binary-rational S of the form $\frac{n}{2^{-k}}$ somewhere inside interval I. Then consider $Left(\hat{S0})$, that is, the predecessor to $\hat{S0}$. This is the binary rational $S - 2^{-(k+1)}$. Call it Z.

If x > Z, then the string we are looking for is $S0$.

If x < Z, then the string we are looking for is $Left(\hat{S})1$: this maps to the interval [Z,Z+2^{k+1}).

The best way to think of this is in terms of the canonical mapping of strings in B^* to intervals in [0,1]. On the top row, a dot over 0; on the second row, a dot over 0 and 0.5; on the third row, a dot over 0, .25, .5, .75, and so on. On the k^{th} row, a dot over $z \times 2^{-k}$. Draw a line from each dot on each row to the dot immediately above it if there is one there, or if there isn't one, to the dot immediately above its left-hand neighbor. When that's done, label the lines below each dot

[30] Brief discussion of Kraft's inequality, which sets necessary and sufficient conditions for there to exist an encoding in B^* (where B is {0,1}), with the prefix property, in which the length of the encodings are $\{l_i\}_i$. The condition is this: $\sum_i 2^{-l_i} \leq 1$. Why?

[31] What is the source of this?

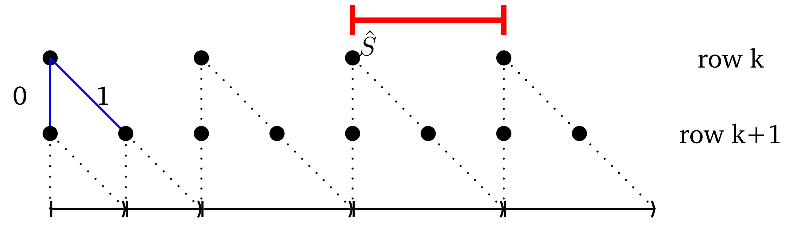


Fig. 3.19: Case 1

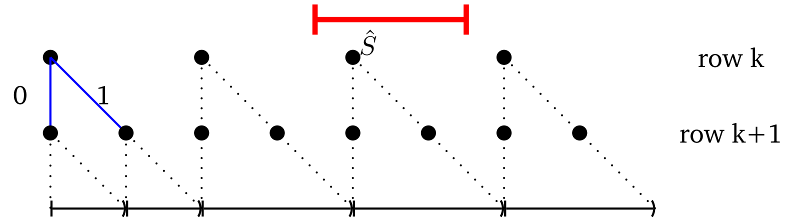


Fig. 3.20: Case 2

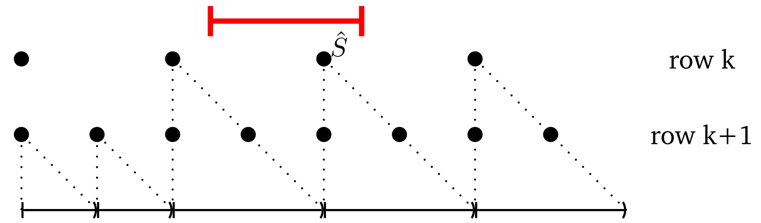


Fig. 3.21: Case 3

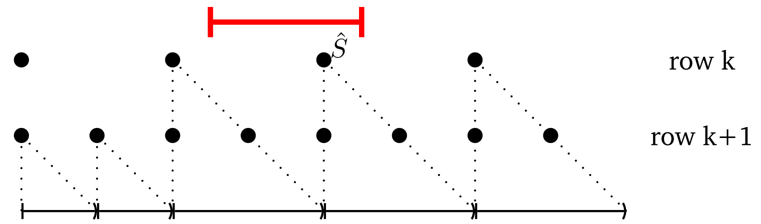


Fig. 3.22: Case 3 bis

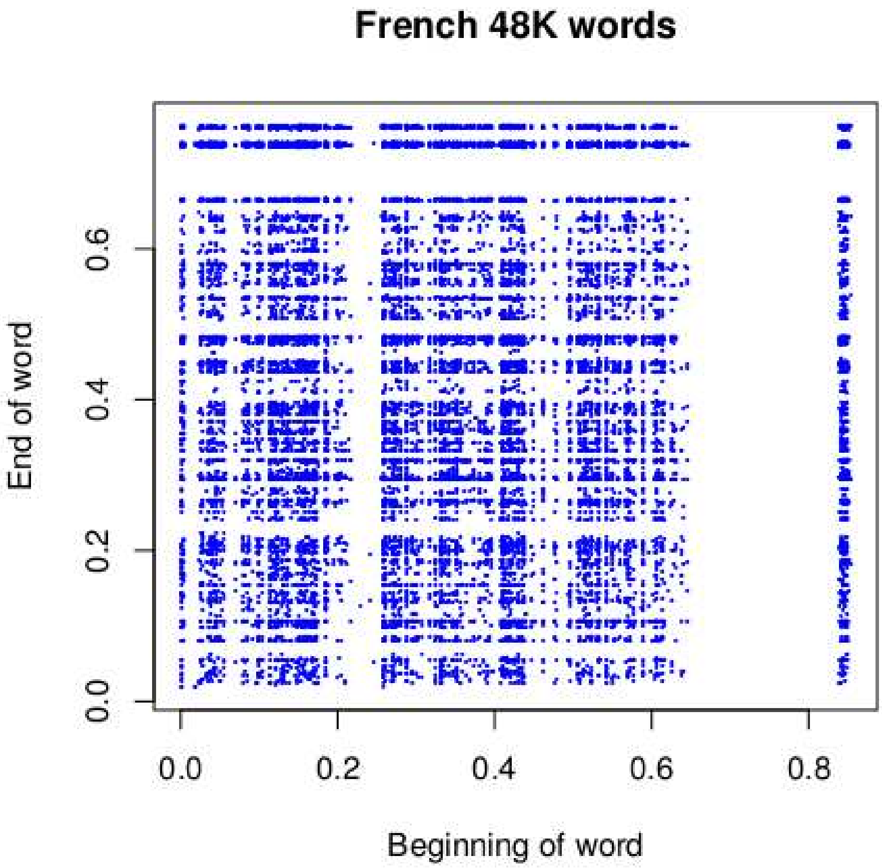


Fig. 3.23: French

'0' and '1', and the concatenation of the labels on the path from the root to each point is a string $s \in B^*$ which uniquely identifies that point: '0.s'.

It should be easy to see that if you have an encoding (in B^*), then it can be embedded into that graph we just made, and the lengths of the intervals corresponding to each of the lowest nodes (the lowest nodes give the encodings) have the property that the sum of their magnitudes is less than 1. The converse is not much harder: if you have a set of $\{l_i\}$, assume that they are sorted in increasing (really, non-decreasing) size, and lay them out, left to right on the picture drawn above, with each l_i on the i^{th} row. Each interval will correspond to a canonical interval, and thus will have an encoding found by tracing down to the node from the root.

This material is well discussed in Li and Vitányi, *An Introduction to Kolmogorov Complexity and its Applications* – the bible of this area. Highly recommended.

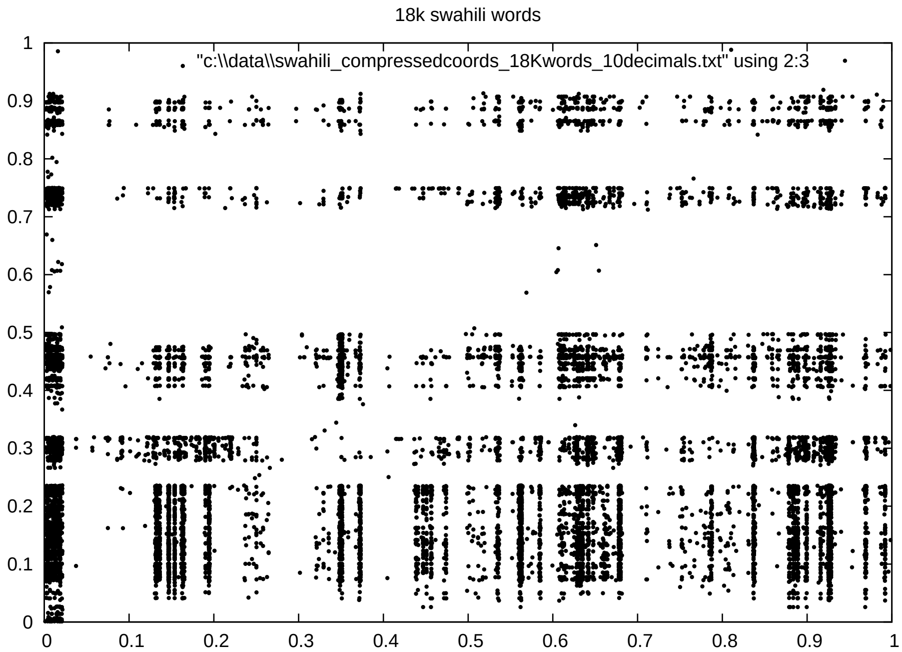


Fig. 3.24: Swahili

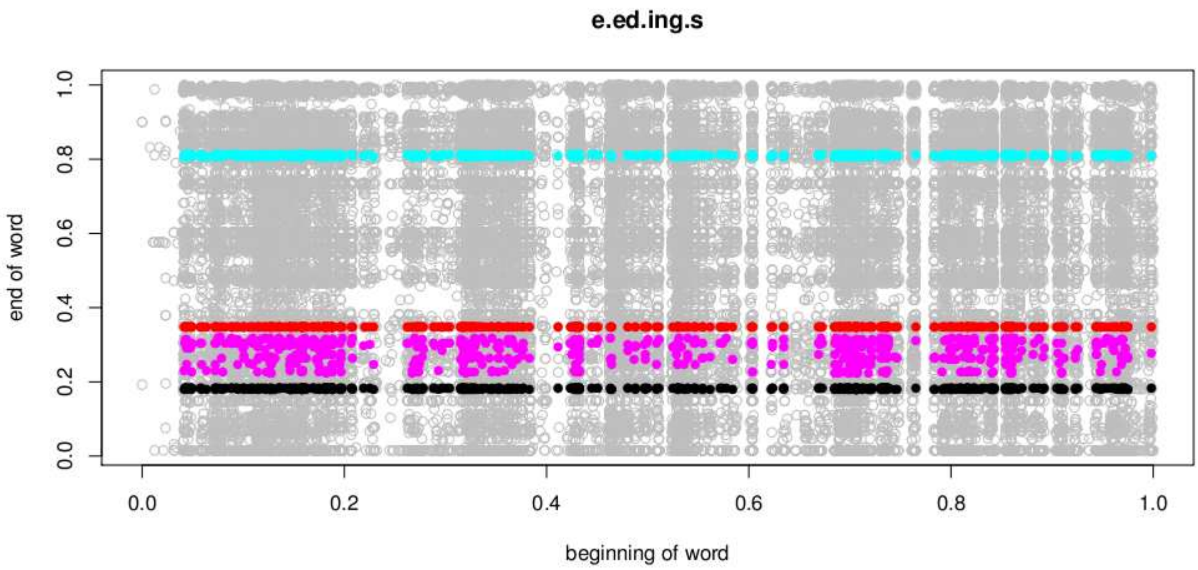


Fig. 3.25: The signature e.ed.ing.s

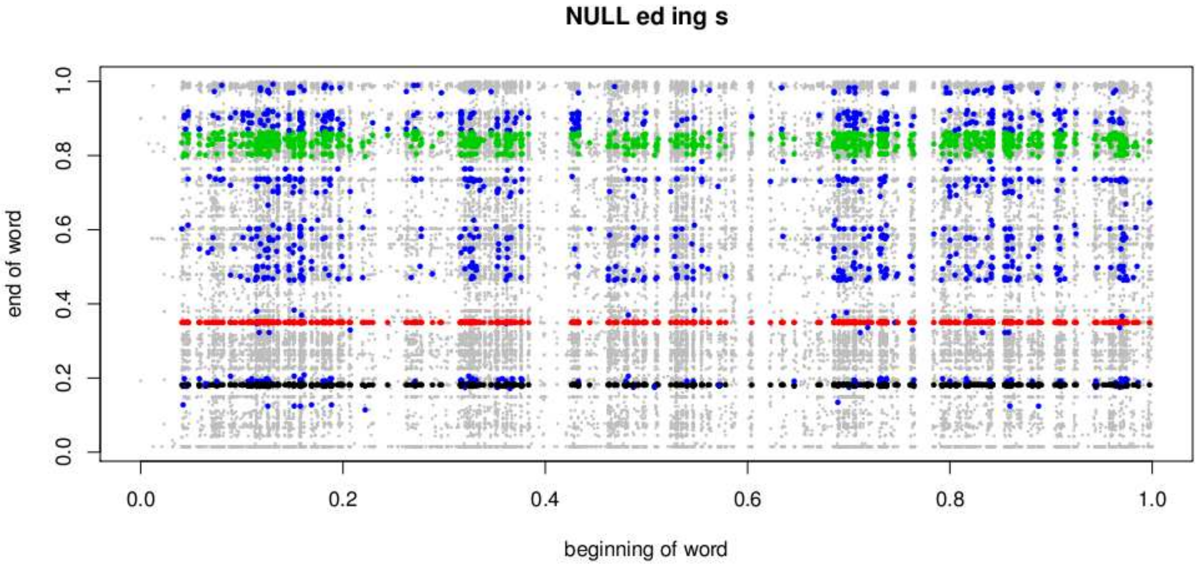


Fig. 3.26: The signature NULL.ed.ing.s

What we have seen so far is a close connection between: (1) probability distributions over alphabets; (2) ways of partitioning [0,1); and (3) identifying intervals I in [0,1) with an encoding from B^* whose length is approximately $-log_2(|I|)$.

Conditional probability and probabilities of strings. We care about assigning distributions to Σ^*. We have already observed that if we assign to a string s a value $Uni(s) = \prod p(s[i])$, then the sum of those values will be 1 if we sum over all the strings of length $|s|$. So we can use this as a probability measure over all strings of a fixed or given length, but certainly not over all of Σ^*!

We have two ways to go. First, we could pick an arbitrary distribution over length λ, and then assign a probability to each string s as $\lambda(|s|) \times Uni(s)$. That would give us a well-formed distribution over all of Σ^*. Second, we could insist that we care not about Σ^*as such, but only about those finite strings in it that end in '#' and have no internal '#'s. Let's call $w = 1 - p(\#)$; it's the probability of emitting a real letter. The probability of emitting the null string followed by # is $1 - w$; then the sum of the probabilities of all strings of the form $x\#$ will be $w(1 - w)$. The sum of the probabilities of all strings of the form $xy\#$ will be $w^2(1 - w)$, sum of the probabilities of all strings of the form $xyz\#$ will be $w^3(1 - w)$, and so on. Those numbers sum to 1, so we're fine: we have a way to assign a distribution, using 'Uni', to strings that end in # and have no internal #s.

Conditional probability: we have a sequence of random variables $U(i)$, but they typically are not independent. For our purposes, we may think of a variable that is independent of what precedes it as being specified by a single distribution labeled with the relevant alphabet Σ, and one that is *not* independent as one that has several such labeled distributions, and the one that is employed is determined by the outcome of a preceding variable—and in the case that we want to consider (the *bigram* model), it is determined by the outcome of the immediately preceding variable.

$$\begin{tikzcd}
Start \arrow[r, "a"] \arrow[dr, "b"'] & A \arrow[r, "a"] \arrow[dr, "a"] & A \arrow[r, "a"] \arrow[dr, "a"] & A \arrow[r, "a"] \arrow[dr, "a"] & A \\
& B \arrow[r, "b"] \arrow[ur, "b"] & B \arrow[r, "b"] \arrow[ur, "b"] & B \arrow[r, "b"] \arrow[ur, "b"] & B
\end{tikzcd}$$

Each *state* is associated with a labeled distribution which is illustrated by its arcs leaving to the right; each random variable has as many outcomes as there are states.

$$pr\,(U(t) = A|U(t-1) = B) = \frac{p(U(t) = A \text{ and } U(t-1) = B)}{p(U(t-1) = B)}$$

This makes perfect sense if we think about using frequencies for our parameters. The probability of h, given that we have just seen a t, is then defined as the probability of a th, divided by the probability of a t.

Last thing: *mutual information*, which we have already talked about. We say $MI(a,b)$ when we really mean something like $MI(U(i) = q, U(i+1) = b)$, for example. In such a case, this is defined as $log\frac{p(U(i)=a)\ \&\ U(i+1)=b)}{p(U(i)=a \times p(U(i+1)=b)}$.

You can see that in such a case, MI(a,b) is $log\frac{p(a|b)}{pr(a)} = log\frac{p(a\ \&\ b)}{p(b)\times pr(a)}$

The upshot of that is simply this: the bigram conditional plog of b, when immediately following a, is equal to b's unigram plog less MI(a,b):

$$plog(U(t) = b \mid U(t-1) = a) = plog(U(t) = b) - MI(U(t-1) = a\ \ \&\ U(t) = b)$$

- Discussion of the next homework problem: finding compounds automatically, and evaluating results.
- Definition of MI(a,b); simple manipulations of the definition of conditional probability.
- MI is the difference between the information content of a corpus using the unigram model and using the bigram model.
- Our goal is to find a sequence of increasingly complex grammars, each of which decreases the plog (=information content) of the data.
- A complex system is one whose entropy, given a zero-order model, is very high, and whose entropy continues to steadily decline over a long sequence of increasingly complex gram-

mars. In a complex system, the complexity does not drop very fast, like a stone – it continues to drop gradually as we strip away more and more regularities within the data.

- Review of evaluating results = Precision, Recall. In both cases, the numerator is the number of correct items your algorithm detected. For precision, the denominator is how many your algorithm detected, whether they were right or wrong. For Recall, the denominator is how many your algorithm should have detected (the number of correct items, according to the Gold Standard).

- Bayes' rule: simple manipulations of the definition of conditional probability. By definition,

$$p(A|B) = \frac{p(A \ \& \ B)}{p(B)}$$

so

$$p(A \ \& \ B) = p(A|B)p(B).$$

and for the very same reason

$$p(A \ \& \ B) = p(B|A)p(A).$$

Hence

$$p(A|B)p(B) = p(B|A)p(A)$$

or

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}.$$

Things start to get tricky when we think of one of the "events" as a *hypothesis*, because we have to ask what we mean by saying that a hypothesis has a particular probability. That is the heart of bayesian reasoning: a willingness to go *there*.

3.17 Chunking: th

Improvement in probability if we start chunking a corpus. State 1: we compute a unigram probability. State 2: we take all occurrences of *th* to form an elementary unit.

$$pr_1(S) = \prod_l \left(\frac{[l]}{N}\right)^{[l]} \tag{3.25}$$

$$pr_2(S) = \prod_{l\in\Sigma, l\neq t,h} \left(\frac{[l]}{N-[th]}\right)^{[l]} \left(\frac{[t]-[th]}{N-[th]}\right)^{[t]-[th]} \left(\frac{[h]-[th]}{N-[th]}\right)^{[h]-[th]} \left(\frac{[th]}{N-[th]}\right)^{[th]} \tag{3.26}$$

$$= \prod_{l\in\Sigma, l\neq t,h} \frac{N^{[l]}}{(N-[th])^{[l]}} \left(\frac{fr_2(t)}{fr_1(t)}\right)^{[t]-[th]} \left(\frac{fr_2(h)}{fr_1(h)}\right)^{[h]-[th]} (fr_2(th))^{[th]} \tag{3.27}$$

$$= \left(\frac{N_1}{N_2}\right)^{|S|-[t]-[h]} \left(\frac{fr_2(t)}{fr_1(t)}\right)^{[t]} \left(\frac{1}{fr_2(t)}\right)^{[th]} \left(\frac{fr_2(h)}{fr_1(h)}\right)^{[h]} \left(\frac{1}{fr_2(h)}\right)^{[th]} (fr_2(th))^{[th]} \tag{3.28}$$

$$= \left(\frac{N_1}{N_2}\right)^{|S|-[t]-[h]} \left(\frac{fr_2(t)}{fr_1(t)}\right)^{[t]} \left(\frac{fr_2(h)}{fr_1(h)}\right)^{[h]} \left(\frac{fr_2(th)}{fr_2(t)fr_2(h)}\right)^{[th]} \tag{3.29}$$

(3.30)

Taking logs, and using $\Delta F = \frac{F_2}{F_1}$:

$$\Delta S = -(|S|-[t]-[h])\Delta N + [t]\Delta fr(t) + [h]\Delta fr(h) + [th] log \frac{fr_2(th)}{fr_2(t)fr_2(h)} \tag{3.31}$$

3.17.0.1 Prose

We will assume throughout our discussion that there is an alphabet Σ in which all raw data is expressed; we could take it to be some version of Unicode, for concreteness's sake. By the term *corpus* we mean a subset of Σ^*; we may refer to it as *data* as well. There is a distinguished symbol in Σ, which we call "space," and represent it either as " " or as "#". Some corpora contain "#" while others do not; we say that the first kind indicate word-boundaries, while the second do not. If we obtain S_2 from S_1 by removing all instances of $\#$ in S_1, then we say that S_2 has been obtained by stripping $\#$ from S_1. We can also speak of the natural lexicon of any corpus that indicates word boundaries in the natural way: after affixing a "#" to the beginning and end of each string in the corpus, we define the lexicon as the set of strings consisting of the maximal substrings of the corpus that do not contain "#".

Information theory is closely related to probability theory and to the theory of encoding. The theory of encoding describes properties of mappings from some universe of formal representations

$\mathcal{L}$ (that might be, for example, the set of sentences of a particular language) to a set $\mathcal{E}$ of strings with very restricted properties: $\mathcal{E}$ might be $\{0, 1\}^*$, for example.[32]

Most of the time, we will want to restrict our attention to cases where $\mathcal{E}$ has the *prefix property*. We say that a set of strings has the prefix property iff there are no pairs of strings S,T in the set such that S is a proper prefix of T. (A string S is a *prefix* of T if $T = S + X$, where "+" is the concatenation operator.)

The reason for this is that it is especially easy to assign probability distributions over such sets.

It is also easy to see (or it *will* be easy to see) that sets of strings S with the prefix property can be associated with a tree, where each $s \in S$ is associated with a terminal element of T.

We will define the information content of an $s \in S$, where $p(s) > 0$, as $-\log p(s) = log\frac{1}{p(s)}$.

Terminology: please bear in mind the difference between *counts*, *frequency*, and *probability*. Counts are numbers (initially, integers) that count the number of occurrences of something. Frequencies are counts which have been normalized, so that the sum of frequencies from an appropriate set will sum to 1.0. Probabilities are parameters of a model. A human being creates a model, and has the privilege if she chooses, to set the parameters however she likes. She may set them to be the same as frequencies, or related in some other fashion to frequencies, but that is a choice.

3.17.1 Important distributions

3.17.1.1 Normal

3.17.2 Bayesian analysis

Bayesian analysis may be defined as the probabilistic analysis of a set of data D (that is, an anaysis which assigns a probability to D) by virtue of selecting a probabilistic model M from a set of possible models $\mathcal{M}$ over which a probability distribution has been defined. That is, we have two entirely different probability distributions at work: we have a distribution over models; we select a specific model m in $\mathcal{M}$, and ask what probability m assigns to our data D.

(Actually, that is a special case, a simple case, of what most real bayesians would expect from a bayesian analysis. They would expect us to consider not a single model m, but rather a distribution d over models. We'll come back to this. For now, we will stick with the special case.)

[32]If you are familiar with probability theory, you might want to know what our measurable sets are. We will restrict our attention to enumerable sets, so all subsets are measurable.

Words

4

4.1 What is a word?

4.2 Word frequencies and Zipf's Law

'

The earliest work on word frequencies is known as Zipf's Law, named after George Zipf. Assume a text composed of words, in the everyday sense. The set of words is the vocabulary V. Some words occur often; others rarely. Let us count the frequency of each word in the text, and then rank the words by their count. Each word w occurs $Count(w)$ times; we will also write this $[w]$ for brevity's sake. Each word w has a rank r_w in the list; if w_1 is more frequent than w_2, then its rank is lower $(r_{w_1} < r_{w_2})$.

The rate at which the frequencies drop off is rather regular, and Zipf's law describes this. A simple version is:

$$freq(w) \times r_w \approx Z_{language}$$

where $Z_language$ is fixed for a given language (though will vary over different languages) and w is a word in the sample from the language. Unless it's important, I will not write the subscript on Z.

This approximation can be rewritten for a particular corpus C:

$$freq(w) = \frac{Count(w)}{|C|} \approx \frac{Z}{r_w}$$

and so we would expect

$$1 = \sum_{w \in V} freq(w) \approx \sum_{w \in V} \frac{Z}{r_w} = Z \sum_{i=1}^{i=|V|} \frac{1}{i}$$

But we know[1] that this sum does not converge as $i \to \infty$, which has made a number of people uncomfortable with this formulation.

[1] and have known since Nicole Oresme, one of the greatest minds of the 14th century, and perhaps of all time, proved it.

To put the point another way, this formula works badly as the size of V gets large. But it also works poorly, from an empirical point of view, when we look at the most frequently words—the most frequent 4 or 5 words. Back in the 1950s, Benoît Mandelbrot proposed a relaxed version of Zipf's law with two additional parameters. One way it which it can be expressed is this, where we clarity things by separating out the normalizing factor:

$$f(k; N, q, s) = \frac{1}{H_{N,q,s}} \frac{1}{(k+q)^s}$$

where

$$H_{N,q,s} = \sum_{i=1}^{N} \frac{1}{(k+q)^s}$$

The reality behind this formula is simpler than it looks at first glance. You can see that if $q = 0$ and $s = 1$ then we are back with the old Zipf's law. So q and s can be looked at as parameters we adjust in order to deal with the problem of the two ends (low rank, high rank) of the curve. Do you see which parameter deals with which end?

http://www.nslij-genetics.org/wli/zipf/

http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html

4.3 Rich get richer

Ijiri and Simon 1975: Explores Bose-Einstein statistics. Suppose we have r (indistinguishable) items in a sequence. They are divided up into n groups, say of adjacent items. We describe this as $(r_1, r_2, \ldots, r_n)$, and $\sum_{k=1}^{n} r_k = r$. Each such description is equiprobable. The main meaning to that is that items within each group are indistinguishable (so they're not multiply counted by permutations), but the groups are distinguishable by order. Thus (0,2), (1,1), and (2,0) each have probability 1/3. If the groups were not distinguishable by order, (1,1) would have probability 1/2. But it doesn't (by the assumption of Bose-Einstein statistics).

Ijiri and Simon show that if consider a process whereby we add an item to this group, and add it to group k with probability r_k/r, then the distribution of groups continues to satisfy Bose-Einstein statistics (though the group is growing, obviously).

They write, roughly, "Let p(i,s) be the probability that a cell will have size i when the aggregate size of all cells is s. Also let p(i) be the steady state probability that a cell will have size i, i.e., p(i) is the limit as s grows to infinity, of p(i,s). Under certain conditions, Gibrat's Law is known to produce as its limiting distribution the Pareto Law, given by $p(i) = Ki^{-\rho}$ in which K and ρ are constant parameters. Under other boundary conditions $\ldots p(i) = M\rho^{-i}$ in which M, ρ are constants, with $\rho > 1$."

These two conditions are the following. Under both, the process increases the number of categories by a constant probability (for them, the probability of a 'bar'), and puts a unit in it. Under one assumption, the new category has exactly one member. Under the other assumption, the new category is created out of an old category, which is split into two parts by the process (and one new unit is added to one or the other). The former leads to a Pareto distribution, the latter to a geometric distribution.

4.4 Yule's characteristics

In *Type-Token Mathematics: A textbook of mathematical linguistics* (1960), Gustav Herdan makes the following point:

Yule's characteristic is defined as

$$K = \frac{S_2 - N}{N^2}$$

where $N = \sum_1^s r n_r$ (by the definition of n_r: it is the number of distinct words occuring with count r) and $S_2 = \sum^s r^2 n_r$.

He then defines K^* as $\frac{S_2}{N^2}$, a quantity close to K. Plugging in the definition of S_2, and defining s as the frequency of the highest frequency word and p_r as the frequency of a word occurring r times (i.e., $\frac{r}{N}$), we find that

$$K^* = \sum \frac{r^2}{N^2} n_r = \sum_{r=1}^{r=s} p_r^2 n_r. \tag{4.1}$$

Now we do the same trick we did before of changing the way we sum over all cases, this time by summing over the individual words in the lexicon. That is, think of each term in the sum in (4.1) as being of the form $p_r[\underbrace{1+1+1\cdots+1}_{n_r\ times}]$, where there is one 1 for each word in the lexicon. So this turns the sum into:

$$K^* = \sum_{w \in lexicon} p_w^2. \tag{4.2}$$

And what is interesting is that we can independently see that this sum is the repeat rate for words: it is the probability that if you pick two words from a corpus, they will be the same word.

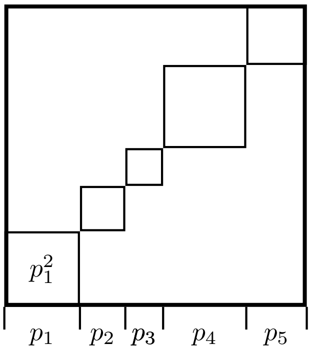
Herdan also points out that

$$K^* - \frac{1}{n} = \frac{S_2}{n^2} - \frac{1}{n} = \frac{n\sum_r r^2 n_r - [\sum_r r n_r]^2}{n[\sum_r r n_r]^2} \tag{4.3}$$

We can change this to frequencies by judicious division by n^3, and define $\pi(r)$ as $\frac{n_r}{n}$, that is, the empirical probability that a word has is of count r:

$$= \frac{\sum r^2\pi(r) - (\sum r\pi(r))^2}{n[\sum_r n\pi(r)]^2} = \frac{\sum_r \sigma_r^2}{\sum_r nM_r^2} \tag{4.4}$$

described as "the coefficient of variation of the mean of the variable r." (88) (and Herdan leaves out the summations in the last expression).



4.4.1 Yule's characteristic K

Herdan thus thinks of K as the variance of the mean number of occurrences per word.

4.4.2 Zipf and Zeta

The zeta (ζ) function just might be the most amazing and beautiful function in all of creation.

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = 1 + \frac{1}{2^s} + \frac{1}{3^s} + \frac{1}{4^s} + \ldots \tag{4.5}$$

We will look at the value of ζ at 1 for a moment. Now, recall that by the prime factorization of integers, any integer has a unique and finite prime factorization. In this section, whenever we use the symbol p, or a slight variant, we mean a $prime$. We use the notation by which p_i is the i^{th} prime number: $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, and so on. Then we can talk about the canonical representation of any integer in terms of the r_i:

$$n = p_{q_1}^{r_1} p_{q_2}^{r_2} ... p_{q_m}^{r_m}$$

for some m, where all the r_i's are > 0, and the q_i's are strictly increasing.

Let's go one step further and write this expansion for any arbitrary n as an infinite product of prime powers, but with the understanding that for a particular n, all but a finite number of the r_i's will be 0 (hence the corresponding prime powers will contribute a factor of 1, which does not matter).

Thus:

$$\frac{1}{n} = \prod_{i=1}^{\infty} \frac{1}{p_i^{r_i}},$$

where the r_i's are all non-negative. So:

$$\zeta(1) = \sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots \tag{4.6}$$

$$= \sum^{\infty} \prod_{i=1}^{\infty} \frac{1}{p_i^{r_i}}, \tag{4.7}$$

Here is where it gets tricky. We will change our original equation (1) from an infinite sum to an infinite product. Every element in the sum can be obtained by picking one power of each of the factors. Once we realize that, then we see we can rewrite this as follows—when you think about the product, think of it as the sum of products made by choosing one element from parenthesized sum:

$$\zeta(1) = \sum^{\infty} \prod_{i=1}^{\infty} \frac{1}{p_i^{r_i}}, \tag{4.8}$$

$$= \left(1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots\right) \left(1 + \frac{1}{3} + \frac{1}{3^2} + \frac{1}{3^3} + \dots\right)$$

$$\left(1 + \frac{1}{5} + \frac{1}{5^2} + \frac{1}{5^3} + \dots\right) \dots$$

Remember that $1 + q + q^2 + q^3 + \dots = \frac{1}{1-q}$. Here, the $\frac{1}{p}$'s 's are the q's. So this product becomes

$$\zeta(1) = \prod_{all\ p_i} \frac{1}{(1 - p_i^{-1})} = \prod_{all\ p_i} \frac{p_i}{p_i - 1}.$$

This does not converge (too bad for Zipf's Law): how could it? It's the product of a lot of numbers all of which are greater than 1. But the zeta (ζ) function (of Euler, and extended to the complex numbers by Riemann) more generally does converge—for example, for real $s > 1$.

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{all\ p_i} \frac{1}{1 - \frac{1}{p_i^s}} \tag{4.9}$$

And (lo! And behold), empirical work on Zipf's law suggests that the best approximation to reality requires a value for s just slightly greater than 1, because the frequencies of all the words has to converge (to 1.0) when we sum up over all the distinct words (i.e., summing over rank position, starting at 1 and going up indefinitely); and Mandelbrot's proposal is that the frequency of the word ranked r in a word list is $\frac{constant}{r+b)^s}$. I don't know what kind of a role the b term plays empirically. But it's clear that it cannot save the sum from diverging when s = 1. We need s to be greater than 1 for the sum to converge.

I think it is very cool that there is some kind of link between linguistics and the most beautiful equation in mathematics. (The Riemann Hypothesis is that the positive zeroes of ζ all have a real part equal to $\frac{1}{2}$.)

4.4.3 Zipf and Pareto

See *Zipf, Power-laws, and Pareto–a ranking tutorial*.
http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html, from which this section is largely drawn.

Zipf's Law: Size of the r^{th} largest word count $\propto r^{-b}$, with b close to 1. Here we take rank to be the independent variable, and its count to be the dependent variable. Bear in mind that saying that a word is of rank n means that there are exactly n words whose count is of that much [whatever it happens to be] *or greater*.

Pareto considered what today we would call the cumulative distribution function: in the case of income, the number of people whose income exceeds a value x. $P[X > x]$ is proportional to x^{-k}. So he is reversing the variables, so to speak: the independent variable (x) is the income, much like the count in Zipf's case; while the dependent variable is much like the rank of Zipf's case.

If we instead consider the probability distribution function (pdf) rather than the cumulative distribution function for Pareto's case of income, we get $P[X = x] \propto x^{-(k+1)}$.

So, back to Zipf: $n \propto r^{-b}$. To flip the axes means essentially to solve this equation for r, i.e., $r \propto n^{-\frac{1}{b}}$

Figure 4.4.1 Many Zipf-type relationships (find source)

Given a Zipfian distribution, the expected count of the r^{th} word is proportional to r^{-b}, i.e., $E[count(w_r)] \propto C_1 r^{-b}$. Paraphrasing this, $P[count(x) \geq C_1 r^{-b}]$ is proportional to r: $P[count(x) \geq C_1 r^{-b}] \propto r$, so $P[count(x) \geq C_1 r^{-b}] = C_2 r$.

On the internet, it is sometimes observed that the number of sites visited by x users $\propto Cx^{-a}$ Using the cumulative function is sometimes clearer graphically, and can avoid the need for binning data.

4.5 Maximize (word-probability/phoneme frequency) over the corpus

Replace this section.

What is the relationship between word probability and phoneme probability in a natural language? How can we ask that question in a coherent or meaningful way?

Let's assume the simple unigram model for the phonemes of a language, and the unigram model for the words as well (but over words, not phonemes, of course), and let's ask how the ratio of these two *kinds* of information content

$$Q = \prod_{n=1}^{|Corpus|} \frac{pr_{syntax}(w_n)}{\prod_{i=1}^{|w_n|} pr_{phono}(w_n[i])} \tag{4.10}$$

4.6 Word discovery

There are two broad families of ways in which we analyze the structure of strings, as we find in data: probabilistic (markov) models, which tell us about the probabilities of selection of elements from Σ in the future, given the past; and segmentation models, whose purpose is to allow for the restructuring of a string of elements from a fine-grained alphabet (such as Σ) to a coarser alphabet $\mathcal{L}$ which is typically called a *lexicon*; each element $w \in \mathcal{L}$ is associated with an element of Σ^*, its "spell-out"—"associated with" rather than "is," because w may be decorated with other information, including meaning, syntactic category, and so on; but to keep things simple, we may assume that no two elements in a lexicon are associated with the same spell-out. We will always assume that each member of Σ is also a member of $\mathcal{L}$ (roughly speaking, each member of the alphabet is a word). If there is an element w of $\mathcal{L}$ associated with the string *the*, we will write w

as **(the)**, and indicate its associated string as $h($**(the)**$)$ or, when it will not cause confusion, simply as **the**. In short, **(the)** is a member of the lexicon, and it is spelled out as $h($**(the)**$)$, or **the**.

Thus any string s of words formed from our lexicon $\mathcal{L}$ is naturally associated with a string in Σ^* in one of two ways: it is associated in a natural way with a string containing word-boundaries (we call that association $h_{\#}$); and it is also associated with a string that does not contain word-boundaries (by $h_{\varnothing}$). For example, if our lexicon contains the words that we write as **(the)** and **(dog)**, then $h_{\#}((the)(dog)) =$ **the#dog**, while $h_{\varnothing}((the)(dog)) =$ **thedog**. We have defined things in this way so that we can be sure that $h_{\#}$ has a well-defined and unique inverse: any string that indicates word-boundaries is associated with a unique string of words. On the other hand, a string that does not indicate word-boundaries will typically be associated with several different strings of words. For example, **the** is associated with **(t)(he)**, **(the)**, and **(t)(h)(e)**, under usual assumptions regarding the lexicon of English.

The *first* problem of word-segmentation, then, is to find a method to undo the stripping of #, the following sense. Given *any* corpus C containing #s, we construct its natural lexicon L and C's stripped version C'. We wish to find a completely general algorithm $\mathcal{S}_1(L, C')$ that can reconstruct C, given the natural lexicon L, and possibly some statistical information available in the original corpus, such as word-frequency and word-sequence information. Needless to say, perhaps, there is no guarantee that such an algorithm exists or that if it exists, it can be found algorithmically. In general, we may wish to develop an algorithm that assigns a probability distribution over possible analyses, allowing for ranking of analyses: given a string *anicecream*, we may develop an algorithm that prefers *an ice cream* to *a nice cream*.

The *second* problem of word-segmentation assumes that the first problem has been solved; the second problem is to find a general algorithm $S_2(C')$ which takes as input a corpus C', which is created by stripping boundaries from a corpus C, and which gives as output a lexicon L which will satisfy the conditions for L established for S_1 in the preceding paragraph. Since there are an astronomical number of different boundary-marked corpora, most with distinct natural lexicons, it goes without saying that if we can solve this problem for naturally occurring corpora, we do not expect it to be extendable to any randomly generable corpus: to put it another way, to the extent that we can solve this problem, it will be by inferring something about the nature of the device that generated the data in the first place.

$$strippedcorpus, lexicon \longrightarrow \boxed{device} \longrightarrow originalcorpus$$
$$strippedcorpus \longrightarrow \boxed{device} \longrightarrow lexicon$$

The problem of word breaking, or word segmentation, may seem artificial from the point of view of someone familiar with reading Western languages: it is the problem of locating the breaks between words in a corpus. In written English, as in many other written languages, the problem is trivial: we mark those breaks with white space. But the problem is not at all trivial in the case of a number of Asian languages, including Chinese and Japanese, where the white space convention is not followed, and the problem is not at all trivial from the point of view continuous speech recognition, or that of the scientific problem of understanding how infants, still incapable of reading, are able to infer the existence of words in the speech they hear around them.

Another computational perspective from which the problem of word breaking is interesting is this: to what extent do methods of analysis that have worked well in non-linguistic domains work well to solve this particular problem? This question is of general interest to the computer scientist, who is interested in a general way regarding the range of problems for which an approach is suitable, and of considerable interest to the linguist, for the following reason. The most important contribution to linguistics of the work of Noam Chomsky since the mid 1950s has been his insistence that some aspects of the structure of natural language are unlearnable, or at the very least unlearned, and that therefore the specification of a human's knowledge of language *prior* to any exposure to linguistic data is a valid and an important task for linguistics. But knowledge of the lexicon of a given language, or the analysis of the words of the lexicon into morphemes, is a most unlikely candidate for any kind of innate knowledge. Few would seriously entertain the idea that our knowledge of the words of this paper, or any other, are matters of innate knowledge or linguistic theory; at best—and this is plausible—the linguist must attempt to shed light on the *process* by which the language learner infers the lexicon, given sufficient data. To say that the *ability* to derive the lexicon from the data is something that few if any would disagree with, and to the extent that a careful study of what it takes to infer a lexicon or a morphology from data provides evidence of an effective statistically-based method of language learning, such work sheds important light on quite general questions of linguistic theory.

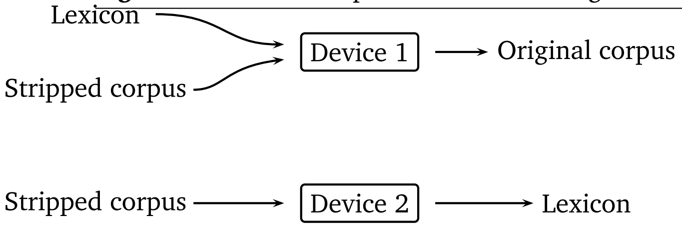
The idea of segmentation of a string $S \in \Sigma^*$ into words is based on a simple intuition: that between two extreme analyses, there must be a happy medium that is optimal. The two extremes here are the two "trivial" ways to slice S into pieces: the first is to not slice it at all, and to leave it as exactly one piece, identical to the original S, while the second is to slice it into many, many pieces, each of which is one symbol in length. The first is too coarse, and the second is too fine, for most strings that are symbolic in any sense at all. The intuition is that there is an intermediate level of "chunking" at which interesting structure emerges, and at which the average length of the chunks is greater than 1, but not enormously greater than 1. The goal is to find the right intermediate level—and to understand what "right" means in such a context.

Another important distinction to bear in mind is that when trying to decide whether a word-break should be placed in a given spot in the string, we can either use current hypotheses about what chunks (i.e., words) exist in the language, or we can use our current hypotheses about what sequences of letters (phonemes) appear most likely inside a word and what sequences occur most likely across different words, i.e., separated by a word-boundary. These two methods are not incompatible, but they are conceptually quite different.

4.6.1 Non-probabilistically: $Sequitur$

Craig Nevill-Manning, along with Ian Witten (see [**?**, **?**]) developed an intriguing non-probabilistic approach to the discovery of hierarchical structure, dubbed *Sequitur*. They propose a style of analysis for a string S, employing context-free phrase-structure rules $\{R_i\}$ that are subject to two very strong restrictions demanding a strong form of non-redundancy: (1) no pair of symbols S, T, in a given order, may appear twice in the set of rules, and (2) every rule is used more than once.

Figure 4.6.1 The two problems of word segmentation



Such sets of rules can be viewed as compressions of the original data which reveal redundancies in the data. An example, taken from [**?**] will make this clear.

Suppose the data is $abcdbcabcd$. The algorithm will begin with a single rule expanding the root symbol S as the first symbol, here a: $S \rightarrow a$. As we scan the next letter, we extend the rule to $S \rightarrow ab$, and then to $S \rightarrow abc$, to $S \rightarrow abcd$, to $S \rightarrow abcdb$, and finally to $S \rightarrow abcdbc$. Now a violation of the first principle has occurred, because bc occurs twice, and the repair strategy invoked is the creating of a non-terminal symbol (we choose to label it 'A') which expands to the twice-used string: $A \rightarrow bc$, which allows us to rewrite our top rule as $S \rightarrow aAdA$, which no longer violates the principles. We continue scanning and extended the top rule, now to: $S \rightarrow aAdAa$, still maintaining the second rule, $A \rightarrow bc$. Scanning the next symbol, b, the top rule becomes $S \rightarrow aAdAab$; and the next, c, the rule becomes $S \rightarrow aAdAabc$. The bc just found is replaced by A, so we have $S \rightarrow aAdAaA$, but this rewrite creates a new violation, since aA appears twice. This leads to the creation of a new non-terminal symbol 'B' and the rule $B \rightarrow aA$, and the top rule is shortened to $S \rightarrow BdAB$. Not surprisingly, the highest level rule is quickly losing its terminal symbols in favor of non-terminal symbols. As the next symbol, d, is scanned, the top rule becomes $S \rightarrow BdABd$, and this triggers a cascade of changes. A new symbol C is created which expands $C \rightarrow Bd$, and the top rule becomes $S \rightarrow CAC$. The rule expanding B (which is $B \rightarrow aA$) is no longer licit, because its only application is to expand the node B which occurs only once in the grammar, in the expansion of the new C. A rule must appear at least twice to survive, so we remove the rule $B \rightarrow aA$ by expanding C in this way: $C \rightarrow aAd$. All the conditions are satisfied, and we have a small hierarchical compression of the original string of symbolic data.

Where are the words, now?

4.6.2 Finding words: Olivier

4.6.3 Minimum Description Length

4.6.3.1 Brent, de Marcken

A good deal of work beginning in the late 1960s. Two widely-cited MIT dissertations in the mid 1990s on this, by Michael Brent and Carl de Marcken.

Let's take the Brown Corpus as our corpus, and following some parts of what de Marcken proposed, establish a simple lexicon consisting of exactly the symbols found in it, and assign each symbol its empirical frequency. We calculate the plog for each sentence, and sum these compressed lengths: we find that there are approximately 16,274,000 bits of information in the Brown Corpus.

Let us gradually add some longer words to the lexicon, by keeping track of which pairs of lexicon members occurred most frequently next to each other. The top 25 most frequent candidates are:

piece	count	status
th	127,717	
he	119,592	
in	86,893	
er	81,899	
an	72,154	
re	67,753	
on	61,275	
es	59,943	
en	55,763	
at	54,216	
ed	52,893	
nt	52,761	
st	52,307	
nd	50,504	
ti	50,253	
to	48,233	
or	47,391	
te	44,280	
ea	41,913	
is	41,159	
ar	40,402	
of	40,296	
ha	39,922	
it	39,304	
ng	39,018	

Now, in each iteration, we find the Viterbi parse (the highest probability parsing, based on a unigram model) and use it. In the second iteration, two words, *the* and *and* are discovered, and an important suffix *ing*, as well as *ic* and *ly*; most of the rest are just small parts of words:

piece	count	status
the	54,598	
ou	35,771	
al	34,471	
and	29,127	
ing	26,520	
as	25,194	
ll	24,681	
ro	22,592	
om	21,070	
ec	20,726	
le	20,269	
ic	20,258	
el	19,661	
me	19,100	
se	17,819	
ly	17,604	
tion	17,339	
em	16,639	
li	16,548	
il	16,523	
co	16,495	
ac	16,072	
wa	14,940	
be	14,907	
ent	14,895	

On the third iteration, some other morphemes and words are proposed; here are about half of the suggestions made on the third iteration:

piece	count	status
for	14,390	word
ofthe	9,899	too large
was	9,455	word
The	9,360	word
no	9,331	word
that	9,273	word
ation	9,188	morpheme
ith	9,164	
ra	9,097	
su	8,813	
lo	8,799	
ol	8,594	
ri	8,561	

On the 50th :

piece	count	status
result	326	word
erial	321	
inwhich	300	too large
understand	297	word
done	296	word
spect	296	morpheme
ger	295	
All	295	word
pract	295	morpheme
close	295	word
complete	295	word
cell	295	word
Nor	294	word
subject	294	word
ionof	294	too large
wind	294	word
edtothe	294	too large
train	294	word
board	293	word
thathe	293	
increas	292	morpheme
ofs	292	too large

On the sixth:

piece	count	status
we	5,598	word
ke	4,330	
you	4,328	word
tothe	4,309	too large
pl	4,243	
man	4,242	word
,the	4,230	too large
not	4,206	word
pre	4,165	morpheme
from	4,146	word
if	4,097	word
ity	4,080	morpheme
ment	3,973	morpheme
them	3,967	word
ate	3,963	word
up	3,916	word
ted	3,854	
so	3,794	word
um	3,776	
mo	3,757	
di	3,723	
ak	3,720	
ard	3,716	
have	3,713	word
edto	3,686	too large

The F ult on County Gr and Ju ry said Fri day an investig ationof Atlan ta 's recent primary election produc ed no evidence that any ir regular ities took place.

A lexicon L is a pair of objects (L, p_L):

- a set $L \in A^*$, and
- a probability distribution p_L that is defined on A^* for which L is the support of p_L. We call L the words.
- We insist that A $\in$ L: all individual letters are words;
- We define a language as a subset of L^*; its members are sentences.
- Each sentence can be uniquely associated with an utterance (an element in A^*) by a mapping F:

Select the lexicon $\mathcal{L}$ which minimizes the description length of the corpus $\mathcal{C}$. A lexicon $\mathcal{L}$ is a distribution $pr_{\mathcal{L}}$ over a subset of Σ^*. $\mathcal{L}$'s length is the length in bits in some specified format (the format matters!) and encoding. Any such distribution assigns a minimal encoding (up to trivial variants) to the corpus, and this encoding requires precisely $-logp(\mathcal{C})$ bits. The description length of a corpus given lexicon $\mathcal{L}$ is defined as $|\mathcal{L}| - logpr_{\mathcal{L}}\mathcal{C}$: select the lexicon that minimizes this quantity (as best you can). $|\mathcal{L}|$ comes into the picture because if we assume $\mathcal{L}$ is expressed in a binary-encoded format in which no morphology is a prefix of another, this encoding induces a natural probability distribution, with $p(l)$ proportional to $2^{|l|}$

4.6.3.2 Other MDL approaches

Minimum Description Length (or MDL) is an approach to data analysis developed by Jorma Rissanen, [**?**, **?**] developing ideas of algorithmic complexity discussed by a range of scholars, notably Solomonoff, Chaitin, Wallace, and notably Kolmogorov. At its heart is the notion that the fundamental challenge of analyzing data is the correct division of a set of observations into information, complexity, and noise, in Rissanen's terminology, each of which can be measured in (Shannon's) *bits*. This terminology is not ideal in the context of applying MDL to the problem of unsupervised language acquisition, because Rissanen's *information* corresponds to the *grammar* that generated the data, the *complexity* is a measure related to the *message* that is encoded by the data, and *noise* is, well, noise.

An MDL approach to the analysis of a set of data D, where $D \subset \Sigma^*$, with Σ an alphabet, begins with an assumption about the class of models $\mathcal{M}$ that will be taken into consideration, and a background assumption about how much encoding, in bits, is required to make any particular grammar completely explicit, typically given in terms of a universal Turing machine. However we choose to define that class of models, each member will be a grammar capable of generating (or accepting) the data D. If we have chosen a model class that contains the grammar g_1(see 1) (generate all strings), then obviously it accepts the data D, but it imposes little structure, and it accepts not only D, but a very, very large and infinite superset of D; this account posits very little *information* (in our terms, very little *grammar*) in the data. We are not obliged to choose so large a model class. Indeed, the artistry that we call science includes the judgment of just what that model class should be.

To repeat: one makes a background assumption about how algorithms will be encoded, and then that assumption allows us to measure the information contained in a grammatical model g that we are entertaining (again, the information is very closely related to the length of the encoding of the grammatical model, or grammar, which we indicate as $|g|$). In addition, we make the assumption that all algorithms are probabilistic. This assumption can be interpreted in two equivalent ways. From the point of view of string accepting, the grammar generates a positive number (< 1) associated with any string in Σ^*; that number is the string's probability; and these probabilities sum to 1.0. From the point of view of generation, any real finite binary sequences of 0's and 1's will be interpreted as a binary fraction beginning "0." (and hence as a rational number between 0 and 1), and a probabilistic grammar can always be interpreted as a device that takes such finite binary expansions, and produces a string; the length of the shortest such binary expansion that generates D is the *complexity* of the message D (written $|D|_g$), and is closely related to the inverse binary logarithm of the probability assigned to the string by an accepting model.

Thus we see that given a model/data pair g/D, we generate two numbers, the information of the model g and the complexity of the data D given the model g. We say that the model/pair then has the *description length* $\mu_{g,D} = |g| + |D|_m$, and we choose a particular model $\hat{g}$:

$$\hat{g} = \arg\min_g \mu_{g,D} = \arg\min_g |g| + |D|_g \tag{4.11}$$

Since a lexicon typically includes its alphabet as a subset (which is to say, each individual letters is also a word, in actual practice), any given model will typically be able to generate a given string D in many different ways (remember the case of $anicecream$), each with its own probability. In practice, we typically define the probability of a string D as the probability associated with one particular *parse* of D.

Putting these threads together, we can see that an MDL approach to word-breaking consists of two things: a suitable definition of a class of lexicon models, where each model assigns a probability distribution over the strings which it generates; and second, a probability distribution over that class of lexicon models. In addition, a model of word-breaking may be linked with a theory of acquisition, which takes a string, its corpus (and hence immediate access to the alphabet Σ in which it is inscribed, so to speak) and proposes a lexicon for it.

Work by Brent [?]; [?]; [?] [?]

4.6.4 Problems with this approach to word discovery

The first set of problems that one encounters in looking at the results of this approach are these: (i) the approach makes pieces that are too large, like *of the*, *to the*, *of course*, etc. (ii) the approach also makes pieces that are too small when the word is relatively infrequent but is composed of pieces that are relatively frequent, like finding *manage ment* as two words, rather than one.

When we put it that way, the problem is obvious. The word-unigram model of language is simply way too simple and simplistic for dealing with natural language. Natural language has an enormous amount of structure, at many different levels, and all that structure is on display in samples of any size from any language. If we expect a model as simple as the word-unigram model to work, we are going to be as sadly disappointed. We need a model as complex as the reality that in fact lies behind the data from the natural languages we look at. We need a model that includes an explicit play for word-internal structure, and an explicit place for word-external structure. We call the first *morphology*, and the second *syntax*.

4.6.5 String edit distance

Goal: find an alignment between two strings which minimizes the "distance" between them. It is possible to customize the definition of "distance" between two strings, but we will consider the default case, which is also called the Levenshtein distance.

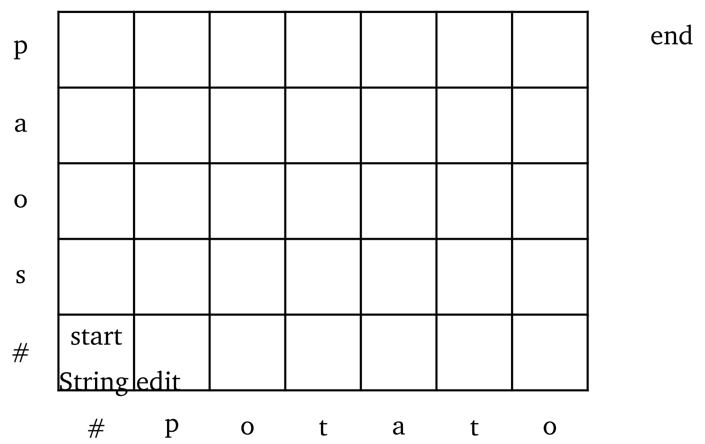
With two words X and Y of length m and n, we set up an m+1 by n+1 array.

Initialization:
for all i and all j, D(i,0) = i and D(0,j) = j.
for i in (1,m):
    for j in (1,n):
        Consider three candidates, and choose the one with the smallest value ("argmin"):
            D(i-1,j) + 1 (add a letter from X that will not be aligned)
            D(i,j-1) + 1 (add a letter from Y that will not be aligned)
            if X[i] = Y[j]: (add a letter from X and one from Y that will be aligned)
                D(i-1,j-1)
            else
                D(i-1,j-1) + 2.
Fill in the entire array, and the minimal distance is D(m,n).

Comparison of strings, both exact and inexact comparison, is an important problem in many computational problems, and it is often useful to be able to give a number which in some sense describes how different two strings are. The best known way to do this involves the string edit distance, which asks essentially the following question. Suppose we set up two strings, S1 and S2, in parallel. The letters in these strings form a set, and imagine all the ways in which these two strings could form a bipartite graph over S1 and S2: that is, a graph in which each edge has one end in S1 and the other in S2—with the further condition that the edges do not cross. A natural term to describe such a graph is as an *alignment* between the two strings. Some pairs of letters are aligned (with the elements of the pair in opposing strings), and any letter not in such a pair is simply unaligned.

Now the crucial step is this: we want to measure how good or bad any alignment is—and typically, we measure how *bad* an alignment is, by providing a measure that gets bigger as fewer pairs of letters are aligned. We will set up the following condition on the formula we use to measure such an alignment. We will say that each unaligned element costs a certain amount, u, and each aligned pair of letters (m,n) costs an amount that is dependent only on what letters m and n are. In a wide range of cases, the choice is made that if m==n, then that cost is 0, and if m != n, the cost is 1. But it is certainly reasonable to have more complicated formula. For example, if our letters are divided into vowels and consonants, then we could establish that the cost of aligning two different vowels or two different consonants was 0.5, but the cost of aligning a vowel and a consonant was 1.

We think about our alignment in two different ways, each of which helps the other. The first way involves a grid:



The familiar symbol # is being used here to mark a null string; we use these rows to indicate the alignment of some letters on one row with nothing at all on the others. Thus the box labeled “start” has the null alignment of nothing with nothing.

It is traditional to start the words in the lower left-hand corner and work up and to the right. Each box (i,j) should be thought of as being the home of an alignment of a substring of S1 and S2: it is an alignment of S1[1:i] and S2[1:j].

We will build it up, and the upper right box will contain the best alignment for the strings S1 and S2. What is surprising is that to find the best alignment for box (i,j), we only need to consider three other boxes: (i-1,j), (i,j-1), and (i-1,j-1). The best alignment for box (i,j) will be a slight modification of one of those three boxes.

To make things really explicit, we will keep two parallel grids of this sort — one for the alignments, and a second one for the *cost* associated with each alignment.

The second way to represent an alignment involves the planar graph we just mentioned:

```
s   o   a   p
|
p   o   t   a   t   o
```

Now we begin with a simple initialization. The best alignments of the lowest row and the left-most column are all "trivial," in the sense that each corresponds to representations with letters on one row but not the other. Box(2,1) corresponds to a null string on the upper string, and the string *p* on the lower string, and for this the best alignment is the only possible alignment: one in which the *t* is aligned with nothing. This costs 2 points. [give table of costs].

Box(3,1) corresponds to a null string on the upper string, and the string *po* on the lower string, and the only possible alignment is with neither letter aligned to anything; the cost is thus 4 points. After carrying out the first 10 initializations, we have two grids like this:

p	soap						
a	soa						
o	so						
s	s						
#		p	po	pot	pota	potat	potato
	#	p	o	t	a	t	o

p	8						
a	6						
o	4						
s	2						
#	0	2	4	6	8	10	12
	#	p	o	t	a	t	o

Now we will fill the chart by rows, proceeding from below to above. For each box (i,j), we consider only three alignments of it:

(i) the alignment in box (i-1,j), to which we add one letter S1[i] but we do not align it with any letters in S2;

(ii) the alignment in box (i,j-1), to which we add one letter S2[j] but we do not align it with any letters in S1; and (iii) the alignment in box (i-1,j-1), to which we add both letters S1[i] and S2[j] and we align them with each other.

4.6.6 Box(1,1)

For box (1,1), there are three options:

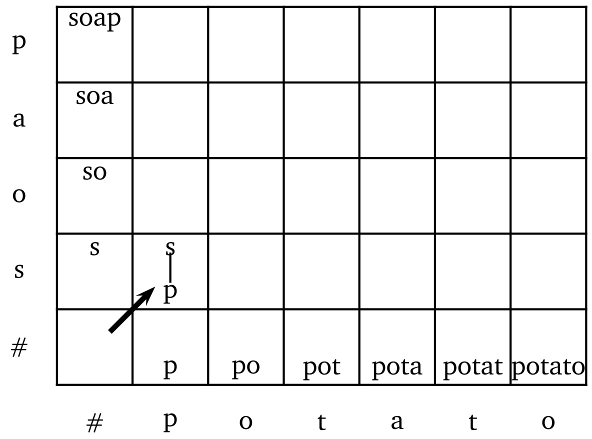


Fig. 4.1: Box (1,1): best alignment

(i) we take the alignment $\left\{ \begin{array}{c} \text{s} \\ \text{-} \end{array} \right\}$ and add the letter p but without aligning them: $\left\{ \begin{array}{c} \text{s} \\ \text{p} \end{array} \right\}$, and this will cost 4 points;

(ii) we take the alignment $\left\{ \begin{array}{c} \text{-} \\ \text{p} \end{array} \right\}$ and add the letter s but without aligning them: $\left\{ \begin{array}{c} \text{s} \\ \text{p} \end{array} \right\}$, and this will cost 4 points;

(iii) we take the alignment $\left\{ \begin{array}{c} \text{-} \\ \text{-} \end{array} \right\}$ and add the letters p and s and align them with each other: $\left\{ \begin{array}{c} \text{s} \\ \text{p} \end{array} \right\}$, and this will cost 2 points.

The third option wins (it costs the least), and so we pick it, and to make it clear which option won, I am going to put an arrow in the picture.

4.6.7 Box(2,1)

Let's do this for box (2,1):

p	8						
a	6						
o	4						
s	2	2					
#	0	2	4	6	8	10	12
	#	p	o	t	a	t	o

Fig. 4.2: Box (1,1): costs

(i) we take the alignment $\left\{\begin{array}{c} \text{s} \\ | \\ \text{p} \end{array}\right\}$ in box(1,1) and add the letter o but without aligning it: $\left\{\begin{array}{c} \text{s} \\ | \\ \text{po} \end{array}\right\}$, and this will cost 4 points (2 from the alignment on the left in box(1,1), and 2 points for the new unaligned o);

(ii) we take the alignment $\left\{\begin{array}{c} \text{-} \\ \text{po} \end{array}\right\}$ and add the letter p but without aligning them: $\left\{\begin{array}{c} \text{s} \\ \text{po} \end{array}\right\}$, and this will cost 6 points (4 from box(2,0) and 2 for the new unanaligned p).

(iii) we take the alignment $\left\{\begin{array}{c} \text{-} \\ \text{p} \end{array}\right\}$ and add the letters p and s and align them with each other: $\left\{\begin{array}{c} \text{s} \\ | \\ \text{po} \end{array}\right\}$, and this will cost 4 points (2 from box (1,0) and 2 for the new aligned pair).

The first and third options receive the same score (they tie), and we have to arbitrarily chose one of them. We will pick the first. This is an arbitrary choice.

Exactly the same reasoning applies for the next four boxes, going to the right, leaving us as we see in the figure for Box (6,1).

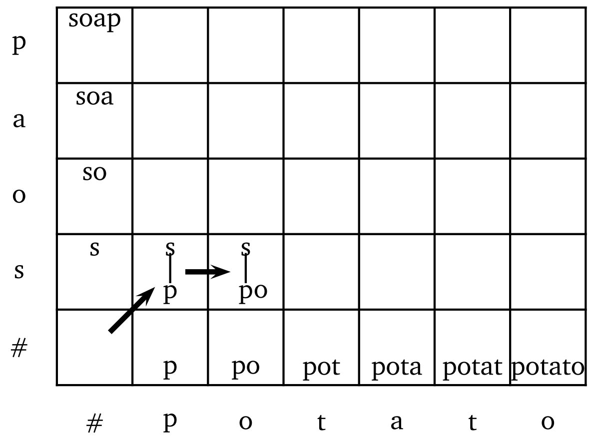


Fig. 4.3: Box (2,1): best alignment

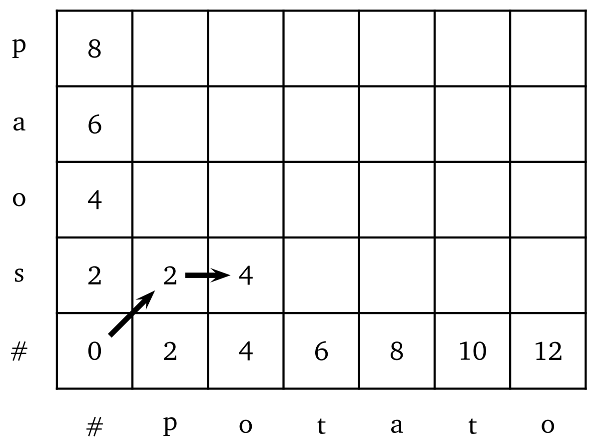


Fig. 4.4: Box (2,1): costs

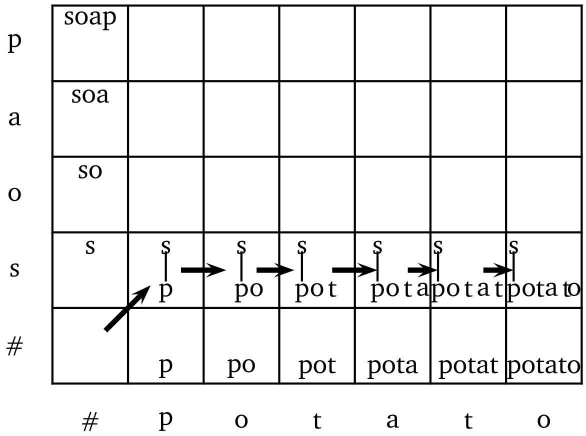


Fig. 4.5: Box (6,1): best alignment

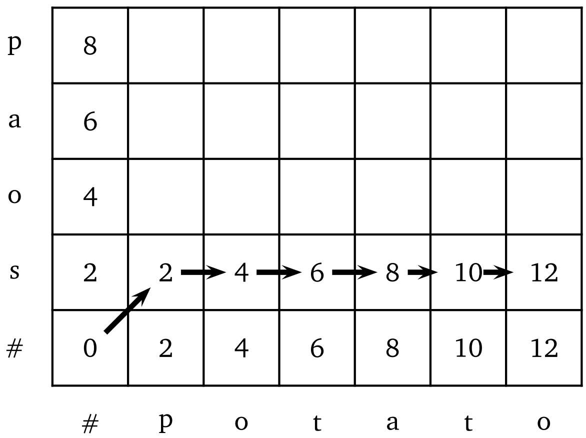


Fig. 4.6: Box (6,1): costs

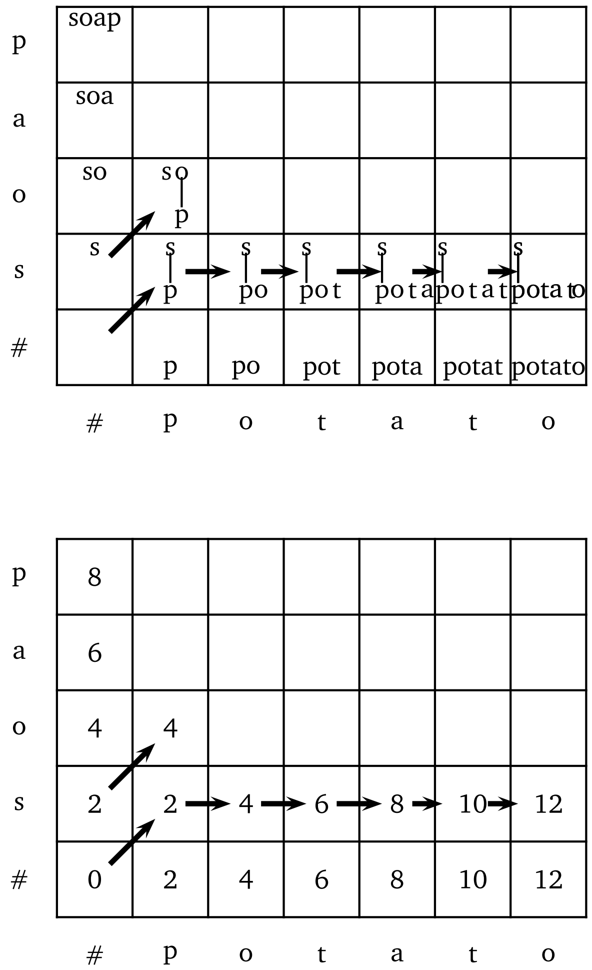


Fig. 4.7: Box (1,2): best alignment and cost

4.6.8 Box (1,2)

Similarly, the analysis for box (1,2) is this; the shift up on the diagonal, aligning the letters p and o, is the winner.

4.6.9 General condition on diagonal arrows in the Box

Any valid alignment is a set of alignments between letters on opposite rows, and any such alignment (i,j) (between S1[i] and S2[j]) corresponds to an arrow pointing diagonally from (i-1,j-1) to (i,j). The non-crossing condition corresponds to the statement that if there is an arrow pointing to (i,j), then if m> i and there is an arrow pointing to (m,n), then n > j.

The string edit distance algorithm constructs arrows pointing *into* each box entry in the way which we have seen. Since there is exactly one arrow pointing into each box, it is possible to uniquely walk back from the end box to the beginning box by going against the sense of the arrows, and this path represents the optimal, least expensive path from start to end.

4.6.10 How do we know it is optimal?

How do we know the path $\mathcal{A}$ constructed by the algorithm is optimal? The short answer is: the final arbiter of which path is the best is chosen by the box marked *end*. There is no way that the *algorithm* can get its path to be chosen by the *end* box if there is an alternative path $\mathcal{B}$ making its way to the end whose total cost is less. It is a *mistake* to think that the algorithm is working its way from bottom-left to upper-right; what is happening is that multiple paths are being created, but the final option is made by the *end* box.

Let us look at the situation for the end box (m,n)—but what we say about the end box will be repeated for all the other boxes as well. The algorithm picks the best path for arriving at (m,n) by looking at the cost of the best path to its three neighbors (m-1,n), (m-1,n-1), and (m,n-1) and incrementing the costs in the relevant ways. If the algorithm can be sure that those three close neighbors really are aware of the least costly path to them, then (m,n) can be sure that there is no other path to get to it: because all paths to (m,n) come from one of those three neighbors.

So we are applying an extension of the familar argument by induction, like this. We easily establish the best path for boxes (0,j) and (i,0) for all values of i and j. And we have just shown that if we know the best path to (m-1,n) and to (m-1,n-1) and to (m,n-1), then the algorithm will necessarily find the best path to (m,n) by chosing the best of the three possible ways of approaching (from below, from the left, or on the diagonal).

And that is all that we need to show, given an appropriate 2-dimensional principle of induction. We want to prove that *the cost of the least costly path from start to end* is equal to the *value computed by the algorithm as indicated*. We'll call this statement $f(i,j)$ for the subbox stretching from start up to position (i,j). $f(i,j)$ is trivially true if i=0 or j=0 (that was our initialization step). And we have already shown that if $f(i-1,j)$ is true, *and* $f(i,j)$ is true *and* $f(i,j-1)$ is true, *then* $f(i,j)$ is true. And those are the conditions that need to hold for the 2-dimensional principle of induction that allows us to conclude that $f(i,j)$ holds for all i,j for which the strings are defined.

If you are still not convinced, then you must worry that there is a best-alignment for a box (i,j) which is not the result of getting there by stepping from (i-1,j) or (i-1,j-1) or (i,j-1). Imagine there is such a best-alignment, which is actually better than the one we have placed in the chart at (i,j). Then look at the alignments of the rightmost letters on each string. If the two rightmost letters are aligned to each other, then remove that pair; if only one of the rightmost letters is aligned, then remove just it. This slightly reduced alignment corresponds to one of those three neighboring spots ((i-1,j) or (i-1,j-1) or (i,j-1)). And it cannot be a better alignment than the one

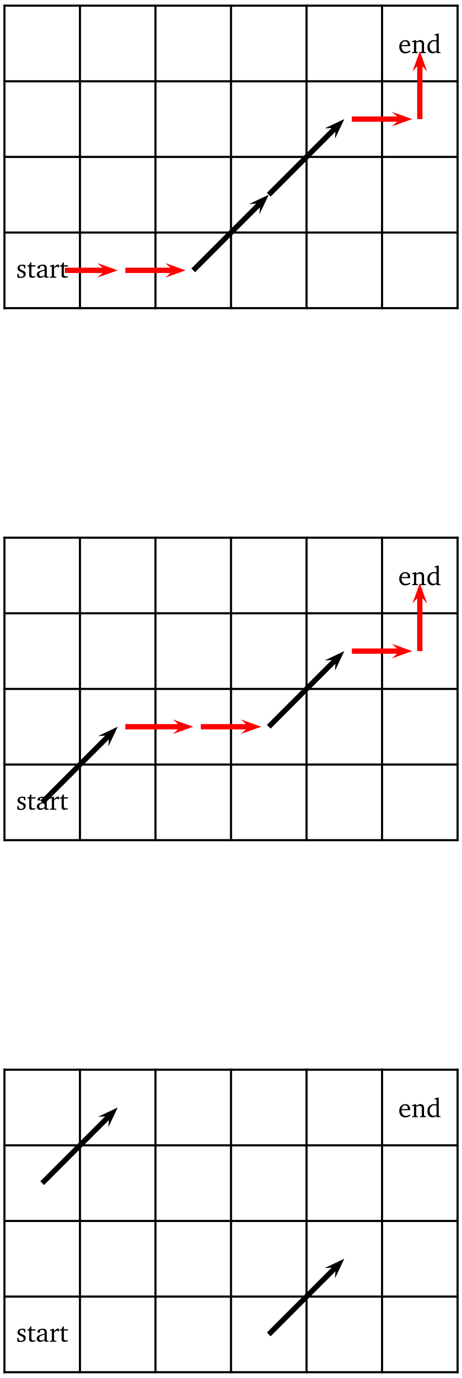


Fig. 4.8: Two sets of alignments that do not violate crossing and one that does

that is already there, by the induction assumption. Therefore, there is no better alignment than the one that can be obtained by arriving at (i,j) from one of the three neighboring boxes.

[Exercise: Show how to reduce the 2-dimensional principle of induction from the regular principle of induction.]

	@	t	h	e	n	a	m	e	o	f	t	h	e	g	a	m	e
@	@:@																
t		t:t															
h			h:h														
e				e:e	*:n	*:a											
r							r:m										
e								e:e	*:o								
s										s:f							
m											m:t						
y												y:h	*:e				
n														n:g			
a															a:a		
m																m:m	
e																	e:e

	@	t	h	e	n	a	m	e	o	f	t	h	e	g	a	m	e
@	0																
t		0															
h			0														
e				0	2	4											
r							4.6										
e								4.6	6.6								
s										7.2							
m											7.8						
y												8.4	10.4				
n														11.0			
a															11.0		
m																11.0	
e																	11.0

Morphology: Making a lexicon

5

5.1 General remarks on morphology

The field of morphology has as its domain the study of internal word structure, and in practice that has meant the study of three relatively autonomous aspects of natural language, which one can identify as morphophonology, morphosyntax, and morphological decomposition. To explain what each covers, we must introduce the notion of *morph*—a natural, but not entirely uncontroversial notion. If we consider the written English words *jump, jumps, jumped*, and *jumping*, we note that they all begin with the string *jump*, and three of them are formed by following *jump* by *s, ed*, or *ing*. When words can be decomposed directly into such pieces, and when the pieces recur in a functionally regular way, we call those pieces *morphs*.

- Morphophonology. It is often the case that two (or more) morphs are similar in form, play a nearly identical role in the language, and each can be analytically understood as the realization of a single abstract element—abstract merely in the sense that it characterizes a particular grammatical function, and abstracts away from one or more changes in spelling or pronunciation. For example, the regular way in which nouns form a plural in English is with a suffixal *-s*, but words ending in *s, sh*, and *ch* form their plurals with a suffixal *-es*. Both *-s* and *-es* are thus morphs in English, and we may consider them as forming a class which we call a *morpheme*: *s, -es* whose grammatical function is to mark plural nouns. The principles that are involved in determining which morph is used as the correct realization of a morpheme in any given case is the responsibility of morphophonology. Morphophonology is, in a real sense, the shared responsibility of the disciplines of phonology and morphology.

- Morphosyntax. Syntax is the domain of language analysis responsible for the analysis of sentence formation, given an account of the words of a language. In the very simplest case, the syntactic structure of a well-formed sentence could conceivably be described as **noun-verb-noun**, where the first noun is the subject and the second the object, but grammar is never that simple; in reality, the morphs that appear in one word (for example, verbal suffixes) may also specify information about the subject or the object (for example, the verbal suffix *-s* in *Sincerity frightens John* specifies that the subject of the verb is grammatically singular). Morphosyntax is the shared responsibility of the disciplines of syntax and morphology.

- Morphological decomposition. While English has many words which contain only a single morpheme (e.g., *while, class, change*), it also has many words that are decomposable into morphs, with one or more suffixes (*help-ful*, *thought-less-ness*), one or more prefixes (*out-last*,) or combinations (*un-help-ful*). But English is rather on the tame side as natural languages go; many languages regularly have several affixes in their nouns, adjectives, and even more often, their verbs. (e.g., Spanish *bon-it-a-s*).

Three interrelated questions:

- Word segmentation: How can we develop a *language-independent* algorithm that takes as input a large sequence of symbols representing letters or phonemes and provides as output that same sequence with an indication of how the sequence is divided into words?
- How can we develop a language-independent algorithm that takes as input a list of words and provides as output a segmentation of the words into morphemes, appropriately labeled as prefix, stem, or suffix—in sum, a morphology of the language that produced the word list?
- How can we implement our knowledge of morphology in computational systems in order to improve performance in natural language processing?

General comments here.

Morphological decomposition. Conversion; compounding.

Inflectional and derivational morphology. A useful distinction is generally made between derivational and inflectional morphology. The distinction falls squarely on whether the phenomenon one is considering is relevant to morphosyntax or not. If it is relevant, then it is considered inflectional morphology, and otherwise it is considered derivational morphology.

Users of natural languages (which is to say, all of us) need no persuasion that words are naturally occurring units. We may quibble as to whether expressions like "of course" should be treated as one word or two, but there is no disagreement about the notion that sentences can be analytically broken down into component words.

In all, or virtually all, languages, it is appropriate to analytically break words down into component pieces, called morphemes; such an analysis is called a morphology, and is the central subject of this chapter. Morphologies are motivated by three considerations: (1) the discovery of regularities and redundancies in the lexicon of a language (such as the pattern in *walk:walks:walking :: jump:jumps:jumping*); (2) the need to predict the occurrences of words not found in a training corpus (e.g.); and (3) the usefulness of breaking words into parts in order to achieve better models for statistical translation and other models particularly sensitive to the meaning of a message.(explain).

Thus morphological models offer a level of segmentation that is typically larger than the individual $letter$, and typically smaller than the $word$. For example, the English word $unhelpful$ can be analyzed as a single word, as a sequence of nine letters, or from a morphological point of view as a sequence of the prefix un, the stem $help$, and the suffix ful.

5.2 Big Picture question

1

Can we build a picture of linguistics in which the goal is to specify a function mapping from the spaces of corpora $\times$ space of grammars such that for a fixed corpus, the optimal value of the function identifies the grammar that is in some *linguistic* sense correct? $g^* = \arg\max -g\, F(C, g)$, where C is a given set of observations ("corpus"), and $g \in \mathcal{G}$: how much is gained by restricting the set $\mathcal{G}$? Such restrictions amount to an assumption about innate knowledge/Univeral Grammar. An alternative strategy is (following Rissanen) to choose a Universal Turing Machine (UTM), and assign a probability to a grammar equal to $2^{-|l(g)|}$, where $|l(g)|$ is the length of the shortest implementation of grammar g on this particular UTM. Does it matter that (1) this statement does not offer any hope that we can recognize the shortest implementation when we see it, or (2) we have no way to choose among UTMs: how do we determine whether UTM-choice matters, in a world of finite data and in which limits may not be taken?

2 If we want to tackle the problem of discovering linguistic structure, both phonology and syntax have the problem that their structure is heavily influenced by the nature of sound and perception (in the case of phonology) and of meaning and logical structure, in the case of syntax. Morphology is less influenced by such matters, and it is possible to emphasize both cross-linguistic variation and formal simplicity. *It is a good test case for language-learning from a computational point of view.*

3 The design of an appropriate objective function—explicating what the description length of a morphology is—is half the project; the other half is designing appropriate and workable discovery heuristics.

4 The goal is not to provide a morphology of English: it is to develop a language-independent morphology learner. Standard orthography (when it departs from phonemic representations) has rules that are similar to (and of the same type, in general) as the rules we find in phonology.

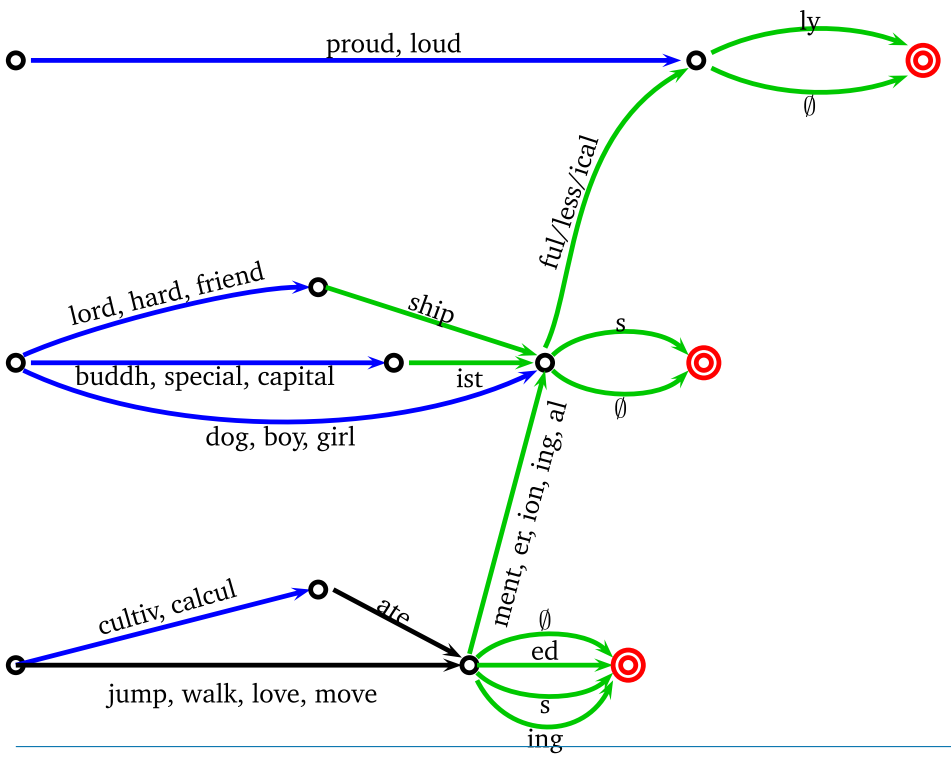


Figure 5.3.1 English morphology: morphemes associated with nodes of an FSA

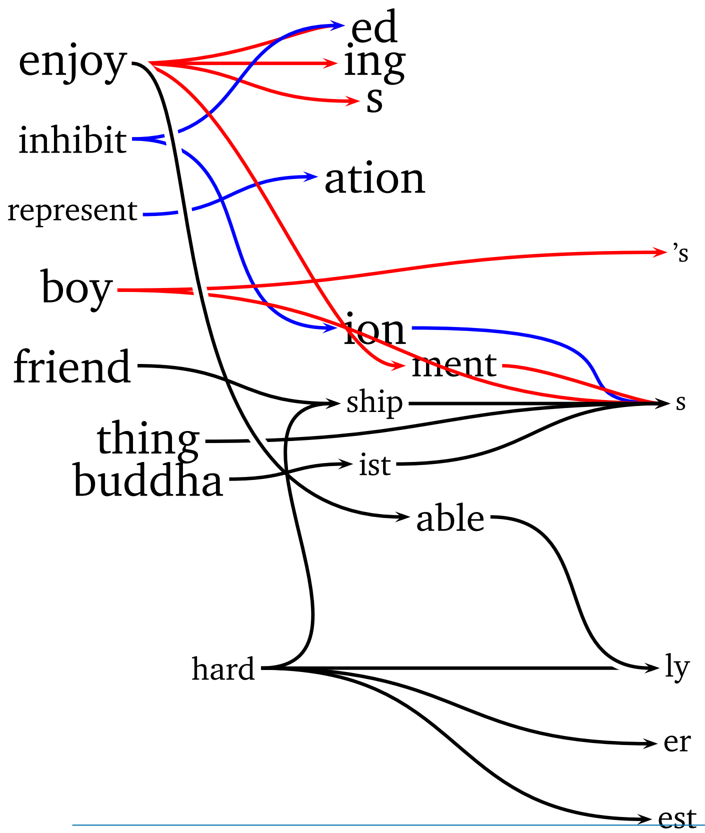
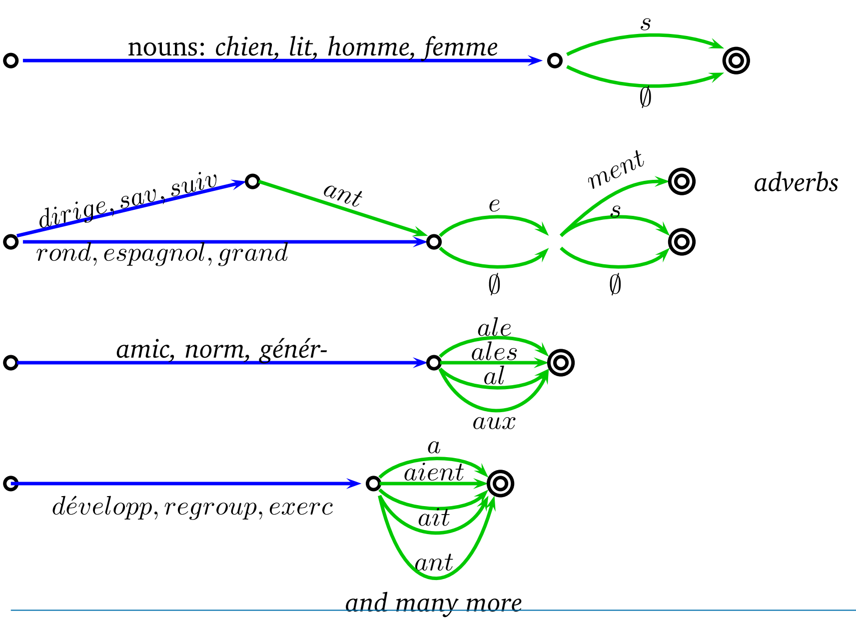
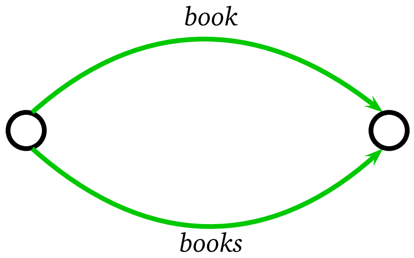


Figure 5.3.2 French



5.3 Morph discovery: breaking words into pieces, and description length of grammar



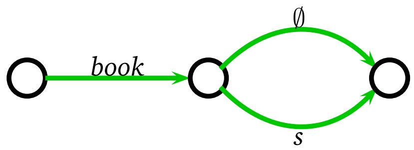
States		Edges				Labels	
number	'pointer to me'	number	states	encoding of states	'pointer to me'	edge ptr.	label
0	0	0	(0,1)	0 1	0	0	book#
1	1	1	(0,1)	0 1	1	1	books#
	2			4	2	2	55
sum	65 bits						

[1] $g^* = \arg\max -g\, F(C, g)$, where C is a given set of observations ("corpus"). Classical MDL offers the joint probability of the data and model as its candidate for F.

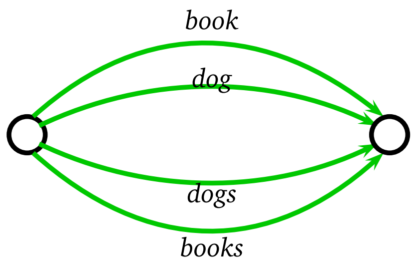
[2] Why **morphology**?

[3] 2 goals: objective function and learning heuristics

[4] Why conventional orthography? Why not phonemes?

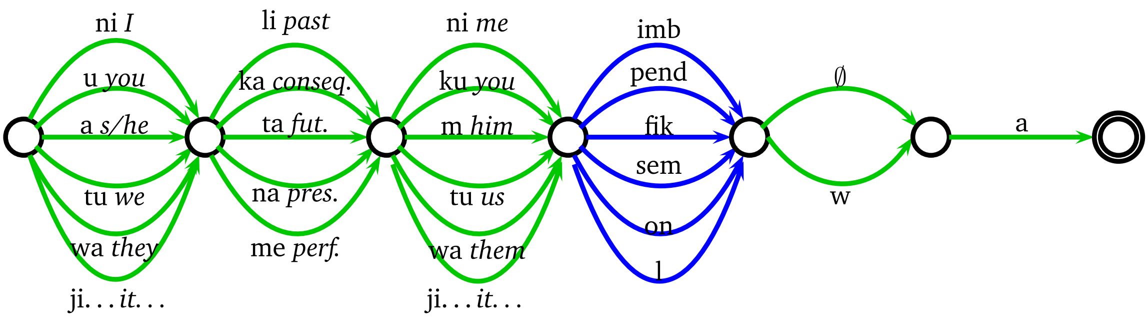


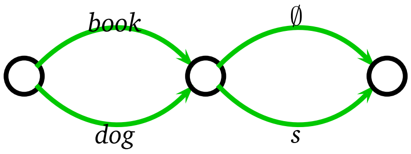
States			Edges					Labels	
number	'pointer to me'		number	states	encoding of states	'pointer to me'		edge ptr.	label
0	0		0	(0,1)	0 10	0		0	book#
1	10		1	(1,2)	10 11	10		10	#
2	11		2	(1,2)	10 11	11		11	s#
	5				11	5		5	40
sum	66 bits								



States			Edges					Labels	
number	'pointer to me'		number	states	encoding of states	'pointer to me'		edge ptr.	label
0	0		0	(0,1)	0 1	00		00	dog#
1	1		1	(0,1)	0 1	01		10	dogs#
			2	(0,1)	0 1	10		10	book#
			3	(0,1)	0 1	11		11	books#
	2				8	8		8	100
sum	126 bits								

Figure 5.3.3 Swahili verbal morphology





States

number	'pointer to me'
0	0
1	10
2	11
	5
sum	95 bits

Edges

number	states	encoding of states	'pointer to me'
0	(0,1)	0 10	00
1	(0,1)	0 10	01
2	(1,2)	10 11	10
3	(1,2)	10 11	11
		14	8

Labels

edge ptr.	label
00	dog#
01	book#
10	#
11	s#
8	60

- How do we choose a morphology (algorithmically)? We want one that endows the data with structure, but not too much. We want to extract redundancy in the data, but not spurious redundancy. In short: how do we find the boundary between real and spurious generalizations regarding word-internal structure?

Figure 5.3.4 Bit cost of signature-based morphology: one particular way to do it (not the only way!)

List of stems:

$$\sum -t \in Stems \sum -i = 1^{|t|+1} - log\, p(t-i|t-i-1)$$

List of affixes:

$$\sum -f \in Affixes \sum -i = 1^{|f|+1} - log\, p(f-i|f-i-1)$$

Signatures:

$$\sum -\sigma \in Signatures \left(\sum -stem\, t \in \sigma - log\, p(t) + \sum -suffix\, f \in \sigma - log\, p(f) \right)$$

Figure 5.3.5 Word probability model: w is $word$, t $stem$, f $suffix$

$p(word) = pr(\sigma - W) * pr(t|\sigma - w) * p(f|\sigma)$,
where word w = stem t + suffix f; each stem belongs to a single signature.
.

Figure 5.3.6 More generally, an acyclic FSA. Natural identity between words and paths through the FSA: $w \approx path - w$. There are various natural, and not so natural, ways to assign these distributions.

PFSA $(\mathcal{V}, \mathcal{E}, \mathcal{L})$, with 4 distributions:
(a) $pr-1(\ )$over $\mathcal{E}$ s.t. $\sum -jpr - 1(e-i, j) = 1$; (b) $pr-2()$ over $\mathcal{V}$;
(c) $pr-3()$ over $\mathcal{L}$ (labels, i.e., morphemes), and
(d) $pr-4()$ over Σ, i.e., the alphabet used for $\mathcal{L}$.
Then $p(w) = p(path-w) = \prod -e \in path - wpr - 1(e).$;
$|FSA| = |\mathcal{V}| + |\mathcal{E}| + |\mathcal{L}|$.
$|\mathcal{V}| = \sum -v \in \mathcal{V}|v|$, where $|v| = -logpr - 2(v)$.
$|\mathcal{E}| = \sum -e \in \mathcal{E}|e|$, where $|e - ij| = |v - i| + |v - j| + |ptr(label - e)|$, and $|ptr(label - e)| = -logpr - 3(label - e)$.
$|\mathcal{L}| = \sum -l \in \mathcal{L}|l|; |l| = -\sum -ilogpr - 4(l - i)$.

- The ideal solution would be one in which we could specify a general function LT ("*linguistic theory*")from pairs of grammar and data to the real numbers: G is the set of all grammars, and D the set of all data. $LT(G, D) \rightarrow Reals$ with the property that

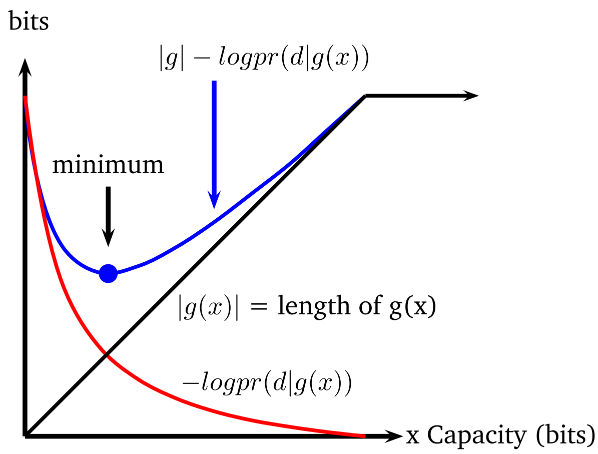
  if $LT(g-1, d) < LT(g-2, d)$, then $g-1$ is a better grammar than $g-2$ for the data d (whatever "better" means to you—this is just a way of saying that it would be ideal if we could write an explicit function to the reals which expresses our grammatical theory's preferences); here, smaller is better, and we are looking for a minimum.

- **Probability** allows an elegant and natural solution. We may elect to choose the grammar which is the *most probable*, given the data (and the technical term here is *maximum likelihood*: roughly speaking, probabilities for theories are really *likelihoods*)

  Find g^* such that g* $= \arg\max -g\ pr(g|d) = \arg\max -gpr(d|g)pr(g)$

Figure 5.3.7 MDL optimization

Interpreting this graph: The x-axis and y-axis both quantities measured in *bits*. The x-axis marks how many bits we are allowed to use to write a grammar to describe the data: the more bits we are allowed, the better our description will be, until the point where we are over-fitting the data. Thus each point along the x-axis represents a possible grammar-length; but for any given length l, we care only about the grammar g that assigns the highest probability to the data, i.e., the $best$ grammar. The red line indicates how many bits of data are left unexplained by the grammar, a quantity which is equal to -1 * log probability of the data as assigned by the grammar. The blue line shows the sum of these two qunantities (which is the conditional *description length* of the data). The black line gives the length of the grammar.



So to use this, we need to

1. specify that our grammars (which generate data) are *probabilistic*, i.e., every form that is output is assigned a probability, which sums to 1.0 over the infinite class of outputs; and part of our test is what the probability that it assigns to the actual data;

2. we need to specify what $pr(g)$ means. It needs to be a function that maps all possible grammars to reals between 0 and 1, and the (infinite) sum of these probabilities is 1.0. The most natural way to do this is to require the grammars to be expressed in binary format, and then take the probability of a particular grammar to be $2^{-1*length(g)}$.

If we do this, then we can replace the argmax with an argmin:

Find g^* such that g* $= \arg\min -g$ [length of g - log $probability - g$ of (d)]

This is the proposal of minimum description length (MDL) analysis.

- An MDL solution thus involves (a) a statement of what possible grammars are, how to compute their probabilities and the probabilities that each assigns to any set of data) and (b) a proposal for search: how to we find the best (or nearly the best) grammar g*, given a set of data?

  Bear in mind that we can imagine lots of solutions to problem (b), all associated with the same solution to (a).

- ## Turning this into a linguistic project

Some details first on the MDL model, followed by some time to talk about the search methods.

We can use the term *length* (of something) to mean the *number of bits = amount of information* needed to specify it. Except where indicated, the probability distribution(s) involved are from maximum likelihood models. The *length* of an FSA is the number of bits needed to specify it, and it equals the sum of these things:

1. List of morphemes: assigning the phonological cost of establishing a lean class of morphemes. Avoid redundancy; minimize multiple use identical strings. The probability distribution here is over phonemes (letters).

$$\sum -t \in morphemes \sum -i = 1^{|t|+1} - log\, pr - phono(t - i|t - i - 1)$$

2. List of nodes v: the cost of morpheme classes

$$\sum -v \in Vertices - log\, pr(v)$$

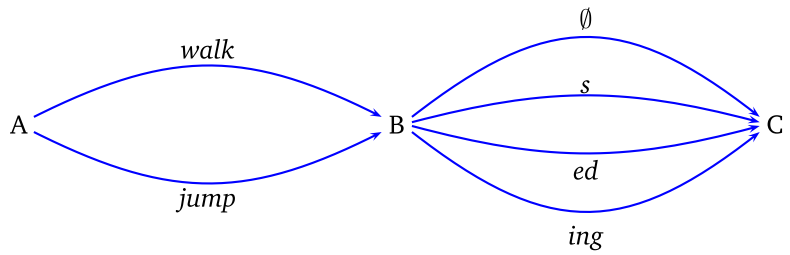
3. List of edges e: the cost of morphological structure: avoid morphological analysis except where it is helpful.

$$\sum -e(v-1, v-2, m) \in\ Edges - log\, pr(v-1) - log\, pr(v-2) - log\, pr(m)$$

(I leave off the specification of the probabilities on the FSA itself, which is also a cost that is specified in bits.)

In addition, a *word* generated by the morphology is the same as a *path* through the FSA. $Pr(w)$ = product of the choice probabilities of for w's path.

So: **for a given corpus, Linguistica seeks the FSA for which the description length of the corpus given the FSA is minimized,** which is something that can be done in an entirely language-independent and unsupervised fashion.



- English suffixes:

  NULL - s - ed - ing - es- er - 's - e - ly - y - al - ers - in - ic - tion - ation - en - ies - ion - able - ity - ness - ous - ate - ent - ment - t (*burnt*) - ism - man - est - ant - ence - ated - ical - ance - tive - ating - less - d (*agreed*) - ted - men - a (*Americana, formul-a/-ate*) - n (*blow/blown*) - ful - or - ive - on - ian - age - ial - o (*command-o, concert-o*) ...

5.4 Linguistica 4 and Linguistica 5

Linguistica 4 and 5 are the two most recent generations of software I have developed with students here to identify morphology automatically. The transition from Lxa 4 to 5 is the biggest transition of any of the changes so far. Let's consider some of the differences.

1. Lxa 4 is written in a superset of C++ called Qt. It can be compiled to run on Windows, Mac OS, and Linux.
   Lxa 5 is written in Python. The older but more developed version 5.0 runs under Python 2.7; the more recent version 5.1 [?] runs under Python 3.4.

2. GUI? Lxa 4 is heavily GUI-centric, which makes it very easy to use. Jackson Lee has written a GUI for Lxa 5.1, but it does not cover all aspects of the program yet.

3. Speed and bulk: Lxa 4 is enormous, just in terms of lines of code, and difficult to hold in a single person's head. It is not well documented. The code does not make a clean distinction between the underlying computations and the GUI, due simply to lack of forethought. Lxa 5 Python is quite small, involving 3 python files, none of which is very large.

4. Lxa 4 uses MDL computation to direct the computation in many ways. This makes it necessary to keep up to date a large number of calculations for each object. This is very messy. Lxa 5 barely uses MDL at all, though it uses robustness as a heuristic, which is a simple calculation that imitates MDL-style analysis.

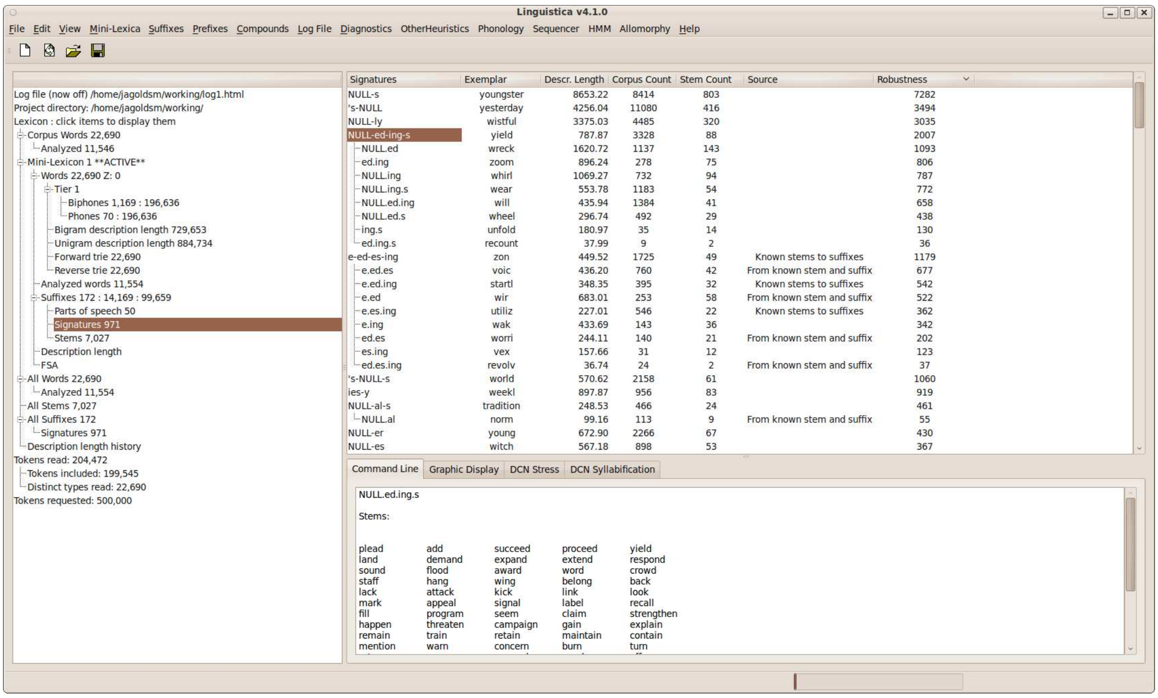
5. Lxa 4 begins with the assumption that a word may be divided into 2 pieces in only one way. Its initial heuristic uses a very restricted subcase of the Harris-criterion. This leads to a frequent inability to deal appropriately with the common cases like *NULL-ped-ping-s*, as with *skip*, and the other parallel cases with consonant doubling. This assumption is deeply embedded in the architecture, which centers around 4 classes of objects: words, stems, affixes, and signatures. (In the code, the class of words is derived from the class of stems, but still.) Each word is associated with a one of each kind of element, with pointers back and forth among them.

   Lxa 5 does not assume that words can be divided in only one way. Its central data structure is a finite state automaton (FSA) which is not deterministic.

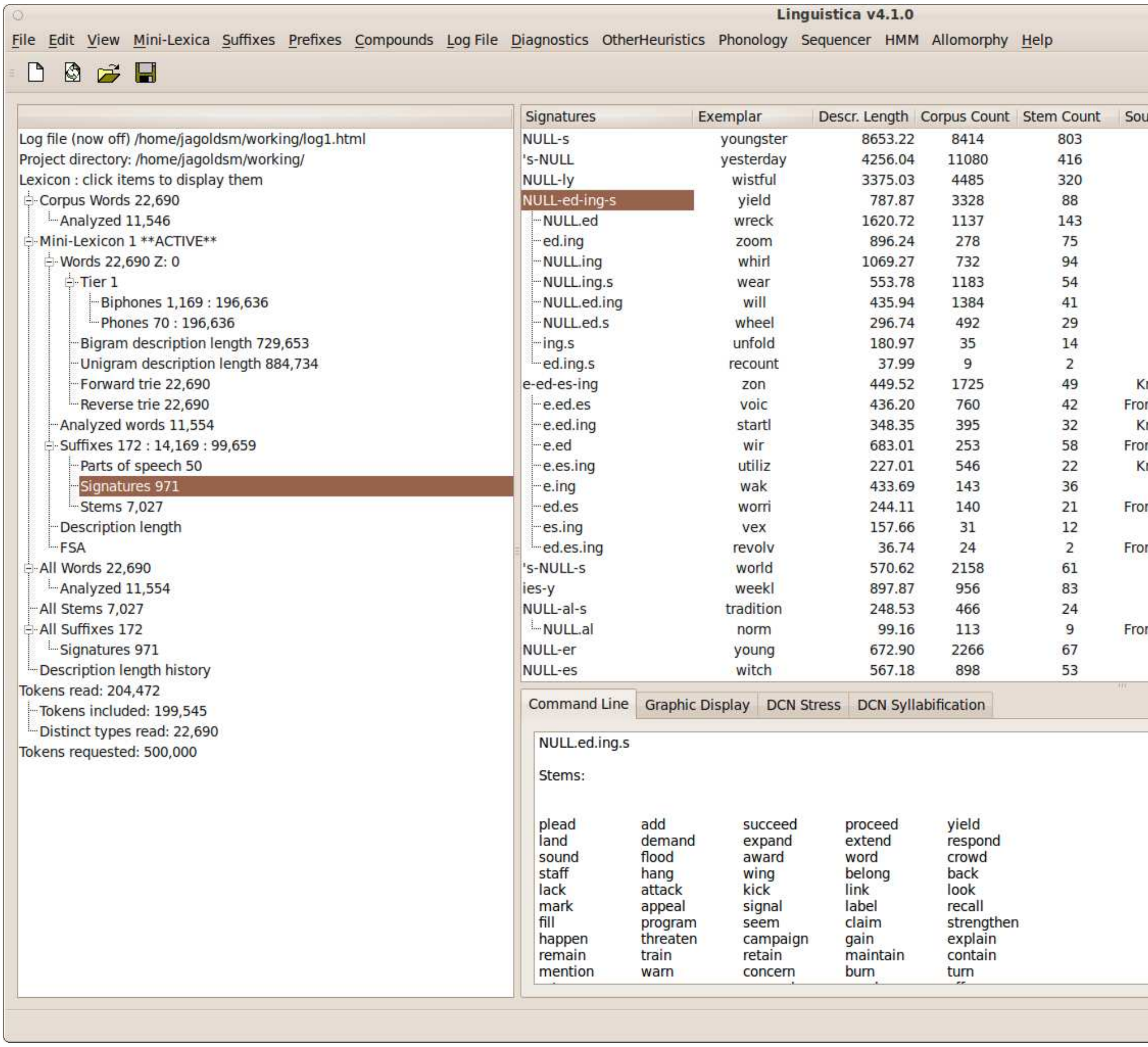
5.4.1 Linguistica 4

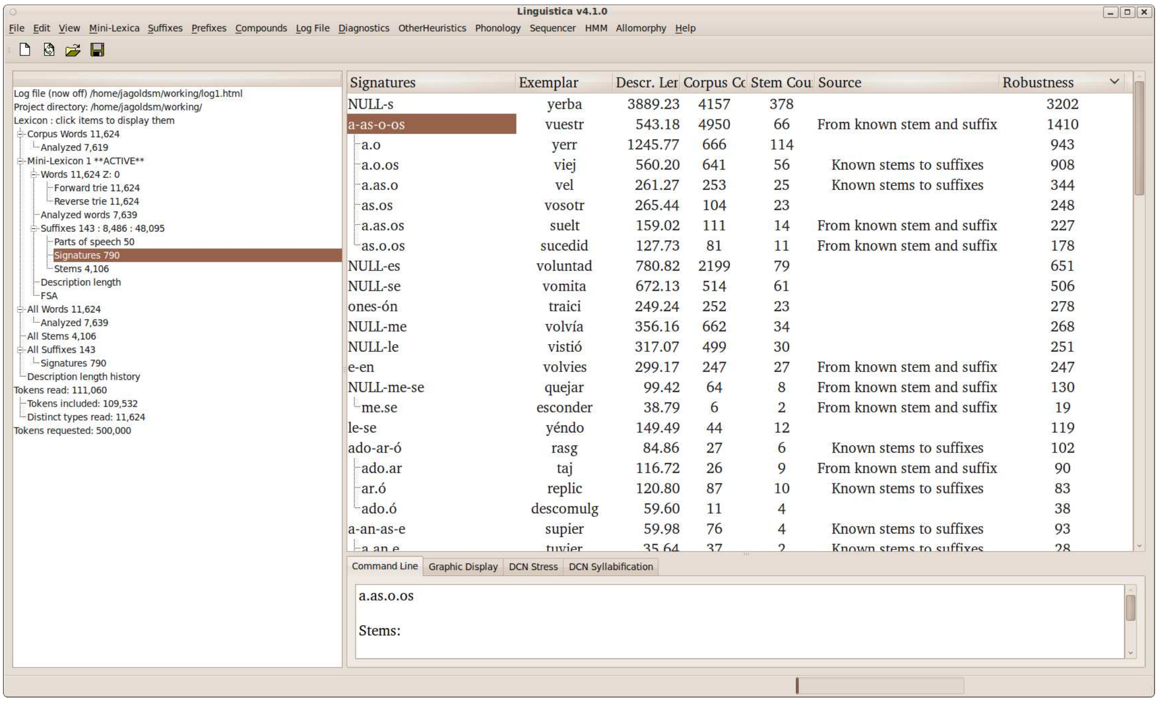
5.4.2 Screenshots

English

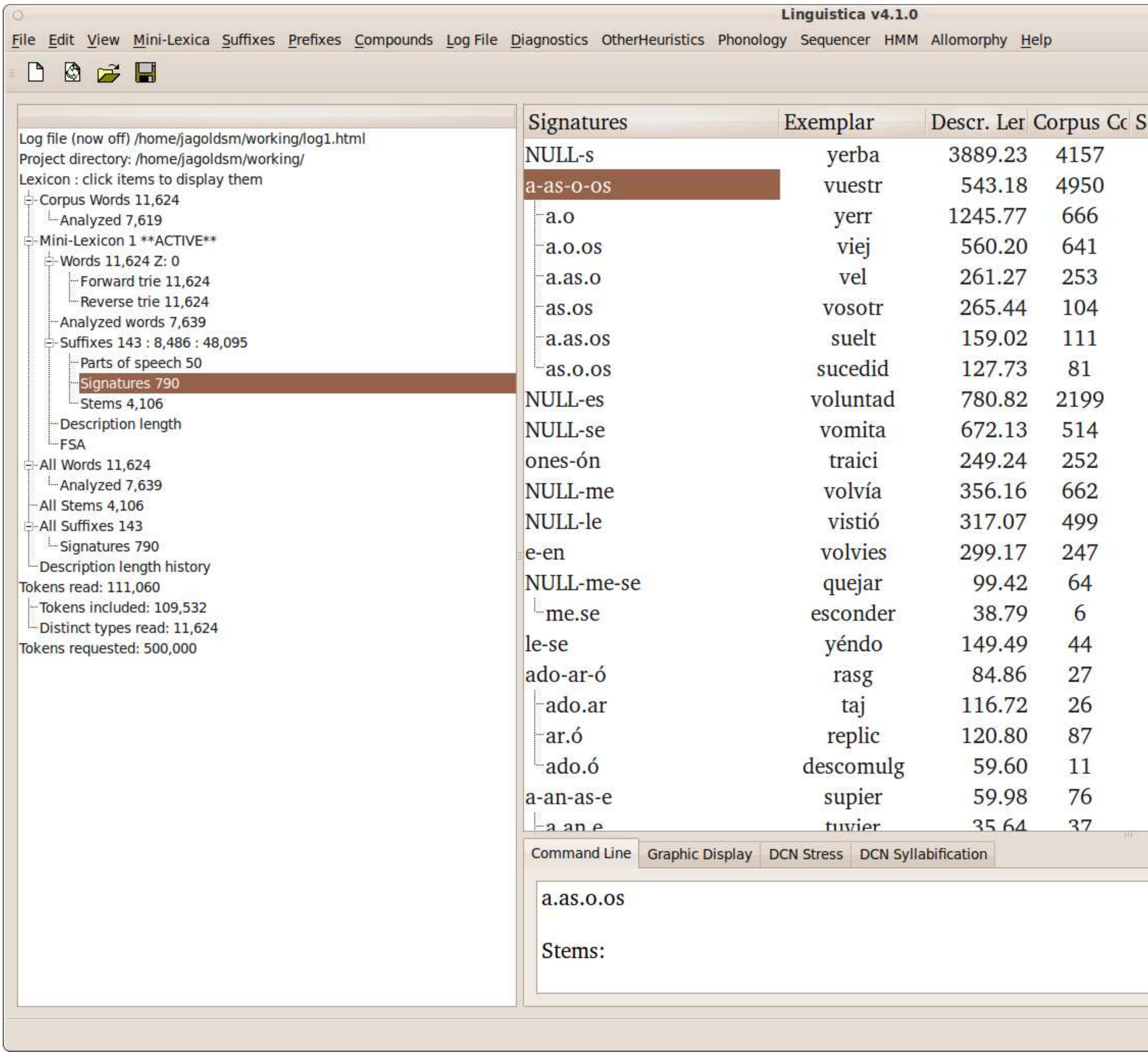


English


Linguistica v4.1.0
File Edit View Mini-Lexica Suffixes Prefixes Compounds Log File Diagnostics OtherHeuristics Phonology Sequencer HMM Allomorphy Help
Log file (now off) /home/jagoldsm/working/log1.html
Project directory: /home/jagoldsm/working/
Lexicon : click items to display them
Corpus Words 22,690
Analyzed 11,546
Mini-Lexicon 1 **ACTIVE**
Words 22,690 Z: 0
Tier 1
Biphones 1,169 : 196,636
Phones 70 : 196,636
Bigram description length 729,653
Unigram description length 884,734
Forward trie 22,690
Reverse trie 22,690
Analyzed words 11,554
Suffixes 172 : 14,169 : 99,659
Parts of speech 50
Signatures 971
Stems 7,027
Description length
FSA
All Words 22,690
Analyzed 11,554
All Stems 7,027
All Suffixes 172
Signatures 971
Description length history
Tokens read: 204,472
Tokens included: 199,545
Distinct types read: 22,690
Tokens requested: 500,000
Signatures Exemplar Descr. Length Corpus Count Stem Count
NULL-s youngster 8653.22 8414 803
's-NULL yesterday 4256.04 11080 416
NULL-ly wistful 3375.03 4485 320
NULL-ed-ing-s yield 787.87 3328 88
NULL.ed wreck 1620.72 1137 143
ed.ing zoom 896.24 278 75
NULL.ing whirl 1069.27 732 94
NULL.ing.s wear 553.78 1183 54
NULL.ed.ing will 435.94 1384 41
NULL.ed.s wheel 296.74 492 29
ing.s unfold 180.97 35 14
ed.ing.s recount 37.99 9 2
e-ed-es-ing zon 449.52 1725 49
e.ed.es voic 436.20 760 42
e.ed.ing startl 348.35 395 32
e.ed wir 683.01 253 58
e.es.ing utiliz 227.01 546 22
e.ing wak 433.69 143 36
ed.es worri 244.11 140 21
es.ing vex 157.66 31 12
ed.es.ing revolv 36.74 24 2
's-NULL-s world 570.62 2158 61
ies-y weekl 897.87 956 83
NULL-al-s tradition 248.53 466 24
NULL.al norm 99.16 113 9
NULL-er young 672.90 2266 67
NULL-es witch 567.18 898 53
Command Line Graphic Display DCN Stress DCN Syllabification
NULL.ed.ing.s
Stems:
plead land sound staff lack mark fill happen remain mention
add demand flood hang attack appeal program threaten train warn
succeed expand award wing kick signal seem campaign retain concern
proceed extend word belong link label claim gain maintain burn
yield respond crowd back look recall strengthen explain contain turn
English

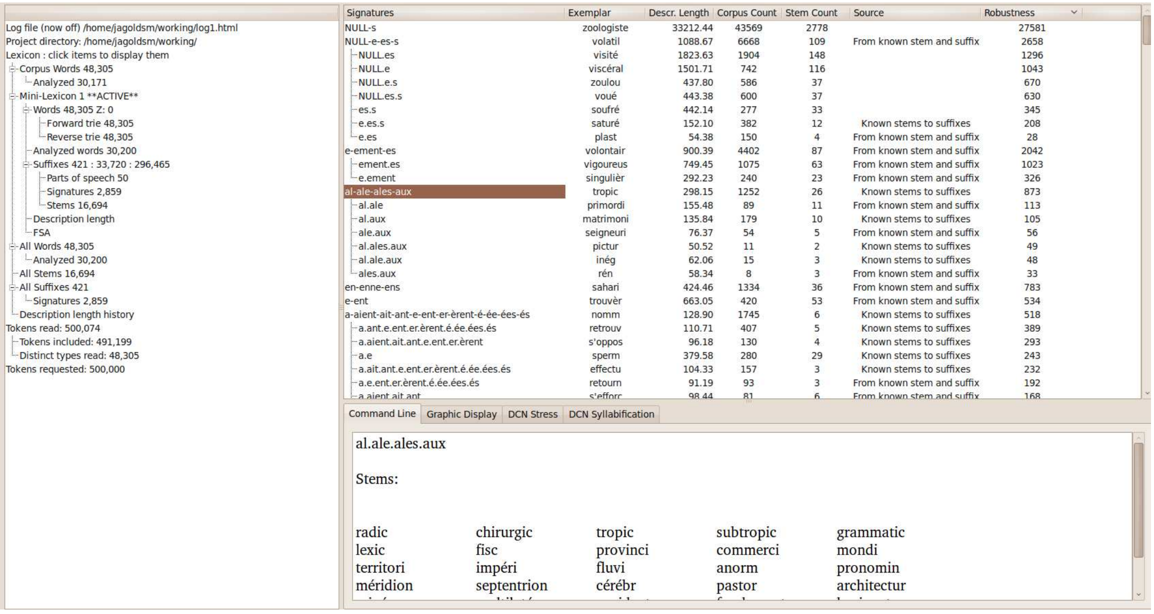

Linguistica v4.1.0
File Edit View Mini-Lexica Suffixes Prefixes Compounds Log File Diagnostics OtherHeuristics Phonology Sequencer HMM Allomorphy Help
Log file (now off) /home/jagoldsm/working/log1.html
Project directory: /home/jagoldsm/working/
Lexicon : click items to display them
Corpus Words 11,624
Analyzed 7,619
Mini-Lexicon 1 **ACTIVE**
Words 11,624 Z: 0
Forward trie 11,624
Reverse trie 11,624
Analyzed words 7,639
Suffixes 143 : 8,486 : 48,095
Parts of speech 50
Signatures 790
Stems 4,106
Description length
FSA
All Words 11,624
Analyzed 7,639
All Stems 4,106
All Suffixes 143
Signatures 790
Description length history
Tokens read: 111,060
Tokens included: 109,532
Distinct types read: 11,624
Tokens requested: 500,000
Signatures | Exemplar | Descr. Ler | Corpus Cc | Stem Cou | Source | Robustness
NULL-s yerba 3889.23 4157 378 3202
a-as-o-os vuestr 543.18 4950 66 From known stem and suffix 1410
a.o yerr 1245.77 666 114 943
a.o.os viej 560.20 641 56 Known stems to suffixes 908
a.as.o vel 261.27 253 25 Known stems to suffixes 344
as.os vosotr 265.44 104 23 248
a.as.os suelt 159.02 111 14 From known stem and suffix 227
as.o.os sucedid 127.73 81 11 From known stem and suffix 178
NULL-es voluntad 780.82 2199 79 651
NULL-se vomita 672.13 514 61 506
ones-ón traici 249.24 252 23 278
NULL-me volvía 356.16 662 34 268
NULL-le vistió 317.07 499 30 251
e-en volvies 299.17 247 27 From known stem and suffix 247
NULL-me-se quejar 99.42 64 8 From known stem and suffix 130
me.se esconder 38.79 6 2 From known stem and suffix 19
le-se yéndo 149.49 44 12 119
ado-ar-ó rasg 84.86 27 6 Known stems to suffixes 102
ado.ar taj 116.72 26 9 From known stem and suffix 90
ar.ó replic 120.80 87 10 Known stems to suffixes 83
ado.ó descomulg 59.60 11 4 38
a-an-as-e supier 59.98 76 4 Known stems to suffixes 93
Command Line | Graphic Display | DCN Stress | DCN Syllabification
a.as.o.os
Stems:

Spanish

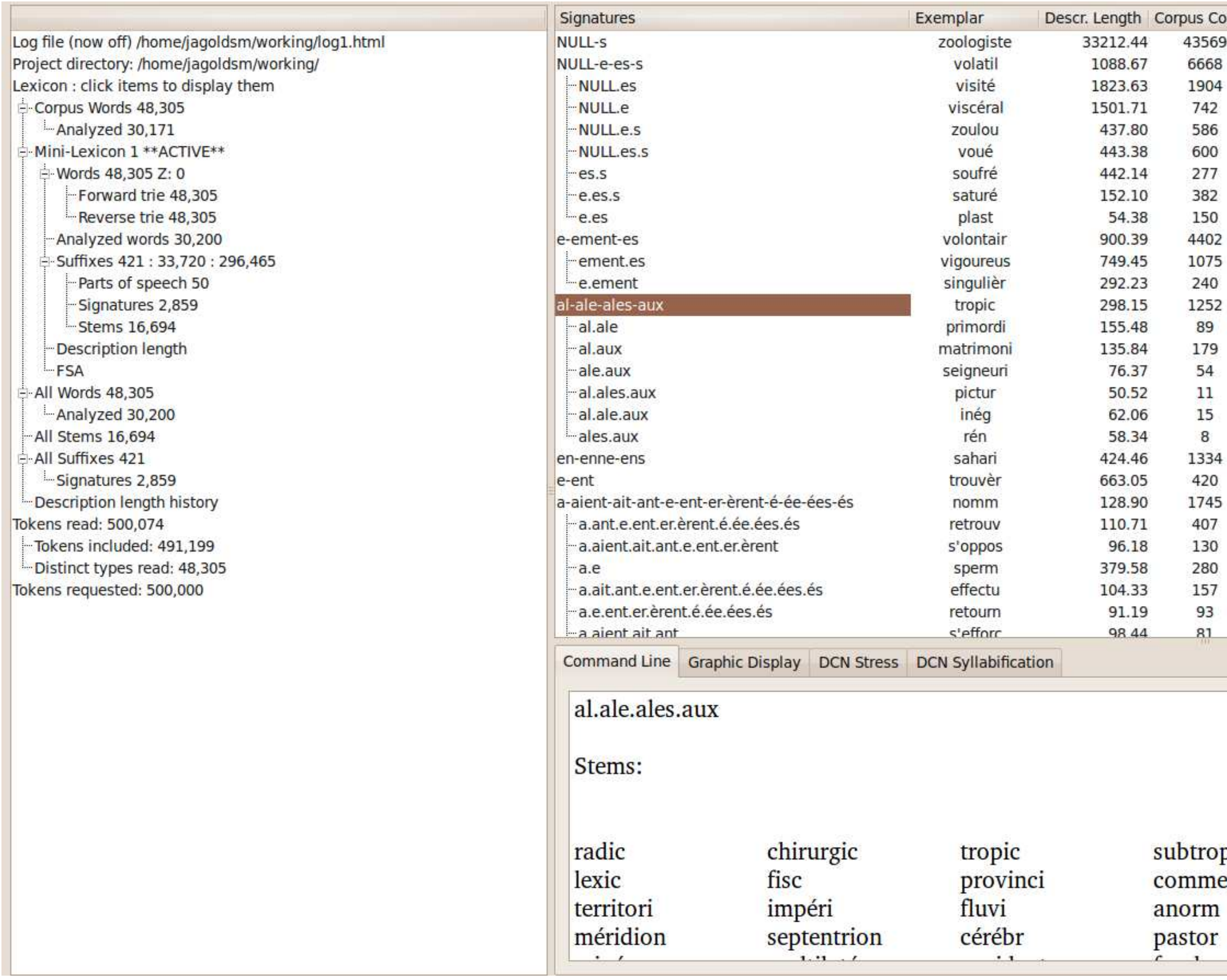

Linguistica v4.1.0
File Edit View Mini-Lexica Suffixes Prefixes Compounds Log File Diagnostics OtherHeuristics Phonology Sequencer HMM Allomorphy Help
Log file (now off) /home/jagoldsm/working/log1.html
Project directory: /home/jagoldsm/working/
Lexicon : click items to display them
Corpus Words 11,624
Analyzed 7,619
Mini-Lexicon 1 **ACTIVE**
Words 11,624 Z: 0
Forward trie 11,624
Reverse trie 11,624
Analyzed words 7,639
Suffixes 143 : 8,486 : 48,095
Parts of speech 50
Signatures 790
Stems 4,106
Description length
FSA
All Words 11,624
Analyzed 7,639
All Stems 4,106
All Suffixes 143
Signatures 790
Description length history
Tokens read: 111,060
Tokens included: 109,532
Distinct types read: 11,624
Tokens requested: 500,000
Signatures Exemplar Descr. Ler Corpus Cc S
NULL-s yerba 3889.23 4157
a-as-o-os vuestr 543.18 4950
a.o yerr 1245.77 666
a.o.os viej 560.20 641
a.as.o vel 261.27 253
as.os vosotr 265.44 104
a.as.os suelt 159.02 111
as.o.os sucedid 127.73 81
NULL-es voluntad 780.82 2199
NULL-se vomita 672.13 514
ones-ón traici 249.24 252
NULL-me volvía 356.16 662
NULL-le vistió 317.07 499
e-en volvies 299.17 247
NULL-me-se quejar 99.42 64
me.se esconder 38.79 6
le-se yéndo 149.49 44
ado-ar-ó rasg 84.86 27
ado.ar taj 116.72 26
ar.ó replic 120.80 87
ado.ó descomulg 59.60 11
a-an-as-e supier 59.98 76
Command Line Graphic Display DCN Stress DCN Syllabification
a.as.o.os
Stems:

Spanish

French

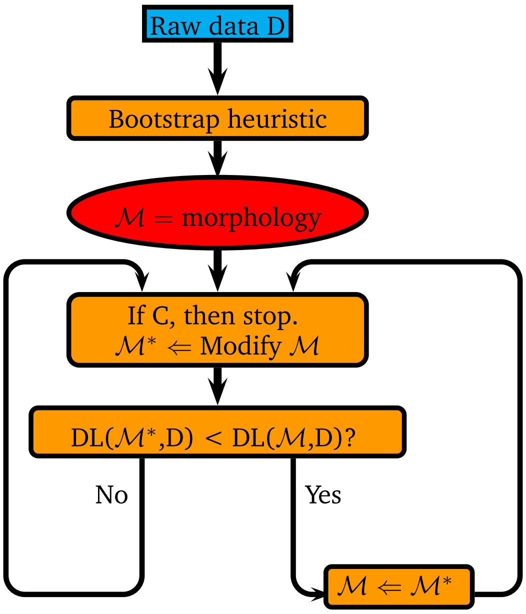


Log file (now off) /home/jagoldsm/working/log1.html
Project directory: /home/jagoldsm/working/
Lexicon : click items to display them
Corpus Words 48,305
Analyzed 30,171
Mini-Lexicon 1 **ACTIVE**
Words 48,305 Z: 0
Forward trie 48,305
Reverse trie 48,305
Analyzed words 30,200
Suffixes 421 : 33,720 : 296,465
Parts of speech 50
Signatures 2,859
Stems 16,694
Description length
FSA
All Words 48,305
Analyzed 30,200
All Stems 16,694
All Suffixes 421
Signatures 2,859
Description length history
Tokens read: 500,074
Tokens included: 491,199
Distinct types read: 48,305
Tokens requested: 500,000
Signatures
Exemplar
Descr. Length
Corpus Count
Stem Count
Source
Robustness
NULL-s zoologiste 33212.44 43569 2778 27581
NULL-e-es-s volatil 1088.67 6668 109 From known stem and suffix 2658
NULL.es visité 1823.63 1904 148 1296
NULL.e viscéral 1501.71 742 116 1043
NULL.e.s zoulou 437.80 586 37 670
NULL.es.s voué 443.38 600 37 630
es.s soufré 442.14 277 33 345
e.es.s saturé 152.10 382 12 Known stems to suffixes 208
e.es plast 54.38 150 4 From known stem and suffix 28
e-ement-es volontair 900.39 4402 87 From known stem and suffix 2042
ement.es vigoureus 749.45 1075 63 From known stem and suffix 1023
e.ement singulièr 292.23 240 23 From known stem and suffix 326
al-ale-ales-aux tropic 298.15 1252 26 Known stems to suffixes 873
al.ale primordi 155.48 89 11 From known stem and suffix 113
al.aux matrimoni 135.84 179 10 Known stems to suffixes 105
ale.aux seigneuri 76.37 54 5 From known stem and suffix 56
al.ales.aux pictur 50.52 11 2 Known stems to suffixes 49
al.ale.aux inég 62.06 15 3 Known stems to suffixes 48
ales.aux rén 58.34 8 3 From known stem and suffix 33
en-enne-ens sahari 424.46 1334 36 From known stem and suffix 783
e-ent trouvèr 663.05 420 53 From known stem and suffix 534
a-aient-ait-ant-e-ent-er-èrent-é-ée-ées-és nomm 128.90 1745 6 Known stems to suffixes 518
a.ant.e.ent.er.èrent.é.ée.ées.és retrouv 110.71 407 5 Known stems to suffixes 389
a.aient.ait.ant.e.ent.er.èrent s'oppos 96.18 130 4 Known stems to suffixes 293
a.e sperm 379.58 280 29 Known stems to suffixes 243
a.ait.ant.e.ent.er.èrent.é.ée.ées.és effectu 104.33 157 3 Known stems to suffixes 232
a.e.ent.er.èrent.é.ée.ées.és retourn 91.19 93 3 From known stem and suffix 192
Command Line
Graphic Display
DCN Stress
DCN Syllabification
al.ale.ales.aux
Stems:
radic chirurgic tropic subtropic grammatic
lexic fisc provinci commerci mondi
territori impéri fluvi anorm pronomin
méridion septentrion cérébr pastor architectur



French

English layers



Lxa 4 as a hill-climbing strategist.

5.4.2.1 Typical output

Lxa 4 can output text files as well. Let's look at the results that it finds as we start with a very small corpus of 1,000 word (types), and double that input several times.

```
{\Large First 1,000 words (distinct words) from the Brown Corpus. }

 Stem Count
 ------------
  38
```

Index	Stem	Confidence	Corpus Count	Affix Count	Affixes
1	elect	SF-1	18	2	ed ion
2	department	From-sigs-find-stems	11	2	's NULL
3	georgia	SF-1	8	2	's NULL
4	vot	SF-1	8	3	e ed ing
5	atlanta	SF-1	7	2	's NULL
6	mayor	SF-1	7	2	's NULL
7	mill	SF-1	7	1	ion
8	new	From-sigs-find-stems	7	2	NULL ly
9	work	From-sigs-find-stems	7	2	NULL ed
10	ask	From-sigs-find-stems	5	3	NULL ed ing
11	court	From-sigs-find-stems	5	2	's NULL

12	daniel	SF-1	5	2	’s NULL
13	operat	From-sigs-find-stems	5	3	ed ing ion
14	pass	From-sigs-find-stems	5	2	NULL ed
15	recommend	SF-1	5	2	NULL ed
16	year	From-sigs-find-stems	5	3	’s NULL ly
17	attend	SF-1	4	2	NULL ed
18	berry	SF-1	4	2	’s NULL
19	ordinary	SF-1	4	2	’s NULL
20	term	SF-1	4	2	NULL ed
21	caldwell	SF-1	3	2	’s NULL
22	general	SF-1	3	2	NULL ly
23	governor	SF-1	3	2	’s NULL
24	like	SF-1	3	2	NULL ly
25	offer	SF-1	3	2	NULL ed
26	order	SF-1	3	2	NULL ly
27	personal	SF-1	3	2	NULL ly
28	reject	SF-1	3	2	ed ion
29	wife	SF-1	3	2	’s NULL
30	administrat	SF-1	2	1	ion
31	allow	From-sigs-find-stems	2	2	NULL ed
32	byrd	SF-1	2	2	’s NULL
33	distribut	SF-1	2	2	e ion
34	effect	SF-1	2	2	NULL ed
35	investigat	SF-1	2	2	e ion
36	protect	SF-1	2	2	NULL ed
37	unanimous	SF-1	2	2	NULL ly
38	consider	SF-1	1	1	ing

```
’s.NULL   12   62
SF1
atlanta berry byrd caldwell court daniel department georgia governor mayor
ordinary wife

NULL.ed   9   34
SF1
allow attend effect offer pass protect recommend term work

NULL.ly   6   21
SF1
general like new order personal unanimous
```

```
ed.ion   2   21
SF1
elect reject

ion   2   9
SF1
administrat mill

e.ed.ing   1   8
Known  stems  to  suffixes
vot

NULL.ed.ing   1   5
Known  stems  to  suffixes
ask

's.NULL.ly   1   5
From  known  stem  and  suffix
year

ed.ing.ion   1   5
From  known  stem  and  suffix
operat

e.ion   2   4
SF1
distribut investigat

ing   1   1
SF1
consider
```

And here is the output from the first 2K words of the Brown Corpus:

```
Stem Count
------------
 460

Index | Stem                 | Confidence          | Corpus Count | Affix Count | Affixes
```

1	state	SF-1	43	3	NULL ment s
2	year	From-sigs-find-stems	34	4	's NULL ly s
3	school	SF-1	33	4	's NULL ing
4	will	From-sigs-find-stems	33	2	NULL ing
5	bill	SF-1	31	4	's NULL ion
6	not	From-sigs-find-stems	27	2	NULL ed
7	hous	From-sigs-find-stems	22	2	e ing
8	elect	SF-1	21	2	ed ion
9	million	SF-1	21	2	NULL s
10	plan	From-sigs-find-stems	20	2	NULL s
11	president	SF-1	20	2	's NULL
12	election	SF-1	18	2	NULL s
13	one	From-sigs-find-stems	18	2	NULL s
14	pay	From-sigs-find-stems	18	3	NULL ing men
15	case	SF-1	17	2	NULL s
16	committee	SF-1	17	2	NULL s
17	other	SF-1	17	2	NULL s
18	court	From-sigs-find-stems	16	3	's NULL s
19	new	From-sigs-find-stems	16	2	NULL ly
20	care	From-sigs-find-stems	15	2	NULL er
21	cost	SF-1	14	3	NULL ly s
22	day	From-sigs-find-stems	14	2	NULL s
23	department	From-sigs-find-stems	14	3	's NULL s
24	grant	From-sigs-find-stems	14	3	NULL ed s
25	program	SF-1	14	2	NULL s
26	home	SF-1	13	2	NULL s
27	vot	From-sigs-find-stems	13	3	e ed ing
28	act	From-sigs-find-stems	12	4	NULL ing ion
29	ask	From-sigs-find-stems	12	4	NULL ed ing
30	bond	SF-1	12	2	NULL s
31	dollar	SF-1	12	2	NULL s
32	fund	From-sigs-find-stems	12	2	NULL s
33	vote	From-sigs-find-stems	12	2	NULL s
34	work	From-sigs-find-stems	12	4	NULL ed er i
35	general	SF-1	11	2	NULL ly
36	hospital	SF-1	11	2	NULL s
37	increas	SF-1	11	3	e ed ing
38	may	From-sigs-find-stems	11	3	NULL er or
39	pass	From-sigs-find-stems	11	3	NULL ed ing
40	depart	NONE	10	1	ment
41	educat	Check-sigs	10	2	ion ional
42	judge	SF-1	10	2	NULL s
43	proposal	SF-1	10	2	NULL s
44	receiv	SF-1	10	3	e ed ing

45		report	SF-1	10	3	NULL ed s
46		unit	From-sigs-find-stems	10	2	NULL ed
47		highway	SF-1	9	2	NULL s
48		kennedy	SF-1	9	2	's NULL
49		law	From-sigs-find-stems	9	2	NULL s
50		legislator	SF-1	9	2	NULL s
51		mak	From-sigs-find-stems	9	2	e ing
52		meet	SF-1	9	2	NULL ing
53		most	SF-1	9	2	NULL ly
54		need	From-sigs-find-stems	9	3	NULL ed s
55		person	From-sigs-find-stems	9	2	NULL s
56		precinct	SF-1	9	2	NULL s
57		rep	From-sigs-find-stems	9	2	NULL s
58		teach	SF-1	9	3	NULL er ing
59		ward	From-sigs-find-stems	9	2	NULL s
60		administrat	SF-1	8	2	ion or
61		georgia	SF-1	8	2	's NULL
62		provid	SF-1	8	3	e ed ing
63		recommend	SF-1	8	3	NULL ation e
64	**	tak	From-sigs-find-stems	8	2	e ing
65	**	take	From-sigs-find-stems	8	2	NULL s
66		time	SF-1	8	2	NULL ly
67		again	SF-1	7	2	NULL st
68		atlanta	From-sigs-find-stems	7	2	's NULL
69		candidate	SF-1	7	2	NULL s
70		constitut	SF-1	7	3	ed ion ional
71		expect	NONE	7	1	ed
72		involv	SF-1	7	2	ed ing
73		legislature	SF-1	7	2	NULL s
74		mayor	SF-1	7	2	's NULL
75		nurs	SF-1	7	2	e ing
76		place	From-sigs-find-stems	7	2	NULL s
77		problem	SF-1	7	2	NULL s
78		republican	SF-1	7	2	NULL s
79		statement	SF-1	7	2	NULL s
80		teacher	From-sigs-find-stems	7	2	NULL s
81		water	SF-1	7	3	NULL ed s
82		william	SF-1	7	2	NULL s
83		action	SF-1	6	2	NULL s
84		aid	From-sigs-find-stems	6	2	NULL s
85		another	SF-1	6	2	's NULL
86		attorney	SF-1	6	2	NULL s
87		bank	SF-1	6	2	NULL s
88		direct	SF-1	6	4	NULL ed ions
89		district	SF-1	6	2	NULL s

90	employ	SF-1	6	3	ed er ment
91	govern	SF-1	6	2	ment or
92	high	From-sigs-find-stems	6	3	NULL er ly
93	lao	From-sigs-find-stems	6	2	NULL s
94	like	SF-1	6	2	NULL ly
95	mean	From-sigs-find-stems	6	2	NULL s
96	meeting	SF-1	6	2	NULL s
97	petition	SF-1	6	2	NULL s
98	plac	From-sigs-find-stems	6	2	e ing
99	propos	NONE	6	1	ed
100	secretary	SF-1	6	2	's NULL
101	system	SF-1	6	2	NULL s
102	approv	NONE	5	1	ed
103	attend	From-sigs-find-stems	5	3	NULL ed ing
104	call	NONE	5	1	ed
105	children	SF-1	5	2	's NULL
106	cotten	SF-1	5	2	's NULL
107	daniel	SF-1	5	2	's NULL
108	doctor	SF-1	5	2	NULL s
109	establish	SF-1	5	2	NULL ment
110	force	From-sigs-find-stems	5	2	NULL s
111	governor	SF-1	5	2	's NULL
112	hear	From-sigs-find-stems	5	2	NULL ing
113	official	SF-1	5	2	NULL s
114	operat	From-sigs-find-stems	5	3	ed ing ion
115	order	SF-1	5	2	NULL ly
116	poll	From-sigs-find-stems	5	2	NULL s
117	receive	From-sigs-find-stems	5	2	NULL s
118	recommendation	SF-1	5	2	NULL s
119	reduc	SF-1	5	3	e ed ing
120	requir	SF-1	5	3	e ed ing
121	road	SF-1	5	2	NULL s
122	robert	SF-1	5	2	NULL s
123	scholarship	SF-1	5	2	NULL s
124	senator	SF-1	5	2	NULL s
125	service	SF-1	5	2	NULL s
126	session	SF-1	5	2	NULL s
127	term	SF-1	5	2	NULL ed
128	add	From-sigs-find-stems	4	2	NULL ed
129	addit	Check-sigs	4	2	ion ional
130	alliance	SF-1	4	2	's NULL
131	amend	SF-1	4	2	ed ment
132	amount	SF-1	4	2	NULL s
133	apparent	SF-1	4	2	NULL ly
134	back	From-sigs-find-stems	4	2	NULL ed

```
 135     berry          SF-1                   4        2  's NULL
 136     boost          SF-1                   4        3  NULL ing s
 137     build          SF-1                   4        2  NULL ing
 138     charg          NONE                   4        1  ed
 139     enforc         SF-1                   4        3  e ed ing
 140     except         SF-1                   4        2  NULL ion
 141     firm           From-sigs-find-stems   4        3  NULL ly s
 142     hearing        SF-1                   4        2  NULL s
 143     hour           From-sigs-find-stems   4        2  NULL s
 144     investigat     SF-1                   4        2  e ion
 145     large          SF-1                   4        2  NULL st
 146     legislat       From-sigs-find-stems   4        2  ion or
 147     list           NONE                   4        1  ed
 148     obtain         SF-1                   4        2  NULL ed
 149     ordinary       SF-1                   4        2  's NULL
 150     personal       SF-1                   4        2  NULL ly

[snip]

 455     unchang        NONE                   1        1  ed
 456     view           NONE                   1        1  ed
 457     validat        NONE                   1        1  ed
 458     want           NONE                   1        1  ed
 459     writ           NONE                   1        1  ing
 460     whipp          NONE                   1        1  ed


 Signature Count
 ---------------
  46

 Signature    Stem Count   Corpus Count
 --------------------------------------
 Remark
 ------
 Stems
 -----

  NULL.s   108   634
  SF1
  action administrator affair aid american amount appointment area attack attorney
  bank believe benefit bond candidate case change committee communist day
  demand dissent district doctor dollar election element event face fee
  force fund gift government hearing highway hold home hospital hour
  individual item judge lao law legislator legislature line matter mean
```

```
meeting method million office official one oppose other outlay part
permit person petition place plan poll portion precinct price principal
problem procedure program project proposal receive recommendation rep republican requirement
right road robert rule saving say scholarship scholastic senator service
session setback site spring statement stay step student system take
teacher texan trouble vote ward week william worker



's.NULL   19   107
SF1
alliance another atlanta berry byrd caldwell children cotten daniel formby
georgia governor kennedy master mayor ordinary president secretary wife

NULL.ed   21   92
SF1
accept add back cover dismiss effect enact end insist interest
nam not obtain offer protect return sound sponsor succeed term
unit

NULL.ly   19   90
SF1
absolute apparent effective final general immediate like main mental most
new order personal previous repeated reported strong time unanimous

e.ed.ing   12   73
Known-stems-to-suffixes
enforc enlarg fac increas liv provid receiv reduc requir rul
serv vot

e.ing   14   72
SF1
authoriz com discharg eliminat handl hous improv licens mak nurs
plac pric ris tak

NULL.ing   11   68
SF1
build enter follow hear lack meet read regard visit will
word

NULL.ed.s   5   43
SF1
grant need report subject water

NULL.ment.s   1   43
```

```
SF1
state

's.NULL.ly.s   1   34
From-known-stem-and-suffix
year

's.NULL.ing.s   1   33
SF1
school

's.NULL.ion.s   1   31
SF1
bill

's.NULL.s   2   30
From-known-stem-and-suffix
court department

ed.ion   3   26
SF1
elect reject revis

NULL.er   5   24
SF1
care few frank off old

ment   11   23
SF1
adjourn advise attach depart develop disappoint ele encourage manage prepay
settle

NULL.ed.ing   4   22
SF1
allow attend learn pass

NULL.ly.s   3   21
SF1
cost firm relative

NULL.ing.ment   1   18
From-known-stem-and-suffix
pay

ed.ing   6   18
```

```
SF1
admitt extend indicat involv permitt warn

NULL.ed.ing.s   2   16
SF1
ask question

NULL.ment   4   16
SF1
achieve establish require retire

ion.ional   2   14
Check-sigs
addit educat

ion.or   3   14
SF1
administrat legislat prosecut

NULL.ed.er.ing   1   12
From-known-stem-and-suffix
work

NULL.ing.ion.s   1   12
From-known-stem-and-suffix
act

e.ion   4   12
SF1
associat distribut investigat violat

NULL.er.or   1   11
Known-stems-to-suffixes
may

NULL.st   2   11
SF1
again large

NULL.ion   3   10
SF1
except port prevent

NULL.er.ing   1   9
From-known-stem-and-suffix
```

```
teach

NULL.ation.ed   1   8
SF1
recommend

NULL.ing.s   2   7
SF1
boost request

ed.ion.ional   1   7
Check-sigs
constitut

ed.ment   2   7
SF1
amend appoint

NULL.ed.ions.or   1   6
SF1
direct

NULL.er.ly   1   6
From-known-stem-and-suffix
high

ed.er.ment   1   6
Known-stems-to-suffixes
employ

ment.or   1   6
SF1
govern

ation.ed   2   6
SF1
inform resign

ed.ing.ion   1   5
From-known-stem-and-suffix
operat

ed.ions   2   4
SF1
discuss select
```

```
ation.ed.ing   1   3
SF1
consult

ation.e   1   2
SF1
realiz
```

5.4.2.2 8K words

```
# Stem Count
# ------------
  2404

# Index | Stem                | Confidence           | Corpus Count | Affix Count | Affixes
# ------------------------------------------------------------------------------------------
  1       that                  From_sigs_find_stems           402             2   's NULL
  2       was                   From_sigs_find_stems           391             2   NULL n't
  3       will                  From_sigs_find_stems           264             2   NULL ing
  4       state                 From_sigs_find_stems           186             5   's NULL d mer
  5       would                 SF_1                           173             2   NULL n't
  6       year                  SF_1                           158             4   's NULL ly s
  7       hav                   From_sigs_find_stems           134             2   e ing
  8       not                   From_sigs_find_stems           130             7   NULL e ed es
  9       new                   From_sigs_find_stems           125             4   NULL ly man s
  10      fir                   From_sigs_find_stems           124             6   e ed es ing r
  11      had                   From_sigs_find_stems           123             2   NULL n't
  12  **  sta                   From_sigs_find_stems           121             3   r te y
  13      one                   From_sigs_find_stems           109             2   NULL s
  14      hom                   Check_sigs                     102             3   e er es
  15      aft                   NONE                            98             1   er
  16      there                 From_sigs_find_stems            95             3   's NULL by
  17 ??   cit                   From_sigs_find_stems            92             5   ation ed es i
  18      other                 SF_1                            92             3   's NULL s
  19      city                  From_sigs_find_stems            84             2   's NULL
  20      oth                   NONE                            80             1   er
  21      school                SF_1                            79             4   's NULL ing s
  22      all                   From_sigs_find_stems            76             4   NULL an ies y
  23      bill                  From_sigs_find_stems            75             5   's NULL ion s
  24      may                   From_sigs_find_stems            74             4   NULL er or s
  25      work                  From_sigs_find_stems            74             7   NULL able ed
```

26	day	From_sigs_find_stems	72	3	’s NULL s
27	play	From_sigs_find_stems	71	7	NULL able ed
28	president	SF_1	71	3	’s NULL ial
29	again	SF_1	70	2	NULL st
30	over	From_sigs_find_stems	70	2	NULL ly
31	game	From_sigs_find_stems	68	3	’s NULL s
32	hous	From_sigs_find_stems	66	4	e ed es ing
33	man	From_sigs_find_stems	66	5	’s NULL or v
34	nation	SF_1	66	4	’s NULL al s
35	some	From_sigs_find_stems	65	2	NULL time
36	tim	From_sigs_find_stems	64	6	NULL e ed es
37	count	From_sigs_find_stems	63	4	NULL ies s y
38	part	From_sigs_find_stems	63	4	NULL ies s y
39	unit	From_sigs_find_stems	61	5	NULL e ed s
40	member	SF_1	60	2	NULL s
41	week	SF_1	60	3	’s NULL s
42	govern	SF_1	59	4	ing ment or
43	administrat	From_sigs_find_stems	57	3	ion ive or
44	night	SF_1	57	4	’s NULL ly s
45	polic	From_sigs_find_stems	56	3	e ies y
46	high	SF_1	55	4	NULL er ly w
47	plan	From_sigs_find_stems	55	5	NULL e es s
48	administration	SF_1	53	2	’s NULL
49	committee	SF_1	53	2	NULL s
50	house	From_sigs_find_stems	52	2	’s NULL
51	meet	From_sigs_find_stems	50	3	NULL ing s
52	miss	From_sigs_find_stems	50	5	NULL ed es i
53	car	From_sigs_find_stems	49	6	’s NULL e r
54	cent	SF_1	49	2	NULL er
55	county	From_sigs_find_stems	49	2	’s NULL
56	off	From_sigs_find_stems	49	3	NULL er ers
57	program	SF_1	49	2	NULL s
58	board	SF_1	48	5	’s NULL ed i
59	call	From_sigs_find_stems	47	5	NULL an ed i
60	club	From_sigs_find_stems	47	3	’s NULL s
61	national	SF_1	47	3	NULL ism ly
62	back	From_sigs_find_stems	46	3	NULL ed s
63	tax	From_sigs_find_stems	46	4	NULL ation e
64	time	From_sigs_find_stems	46	2	NULL ly
65	monday	SF_1	45	2	’s NULL
66	report	SF_1	45	5	NULL ed ers
67	should	SF_1	45	3	NULL er n’t
68 **	elec	Check_sigs	44	3	t ted tion
69	could	From_sigs_find_stems	43	2	NULL n’t
70	government	SF_1	43	4	’s NULL al s

71	most	From_sigs_find_stems	43	2	NULL ly
72	per	From_sigs_find_stems	43	3	NULL son t
73	run	From_sigs_find_stems	43	2	NULL s
74	start	From_sigs_find_stems	43	5	NULL ed er i
75	did	From_sigs_find_stems	42	2	NULL n't
76	form	From_sigs_find_stems	42	6	NULL ally by
77	john	From_sigs_find_stems	42	3	NULL s son
78	tak	From_sigs_find_stems	42	3	e es ing
79	direct	SF_1	41	6	NULL ed ing
80	even	From_sigs_find_stems	41	3	NULL ing t
81	month	From_sigs_find_stems	41	4	's NULL ly s
82	open	From_sigs_find_stems	41	5	NULL ed er i
83 ??	und	NONE	41	1	er
84	court	From_sigs_find_stems	40	3	's NULL s
85	dur	NONE	40	1	ing
86	public	From_sigs_find_stems	40	4	NULL ity ize
87	republican	From_sigs_find_stems	40	3	NULL ism s
88	com	From_sigs_find_stems	39	5	e es ic ing
89	council	SF_1	39	3	's NULL man
90	university	From_sigs_find_stems	39	2	's NULL
91	american	From_sigs_find_stems	38	2	NULL s
92	ask	From_sigs_find_stems	38	4	NULL ed ing
93	election	SF_1	38	2	NULL s
94	party	From_sigs_find_stems	38	2	's NULL
95	pass	From_sigs_find_stems	38	4	NULL ed es i
96	vot	From_sigs_find_stems	38	5	e ed er es i
97	william	SF_1	38	2	NULL s
98	democrat	From_sigs_find_stems	37	3	NULL ic s
99	general	From_sigs_find_stems	37	3	's NULL ly
100	jur	From_sigs_find_stems	37	4	ies ist ors
101	mill	From_sigs_find_stems	37	4	NULL er ion
102	sunday	SF_1	37	2	's NULL
103	case	From_sigs_find_stems	36	4	's NULL s y
104	expect	SF_1	36	4	NULL ations
105	get	From_sigs_find_stems	36	2	NULL s
106	league	SF_1	36	4	's NULL r s
107	rul	From_sigs_find_stems	36	5	e ed ers es
108	universit	From_sigs_find_stems	36	2	ies y
109	yesterday	SF_1	36	2	's NULL
110	ball	From_sigs_find_stems	35	2	NULL s

[snip]

2399	wip	NONE	1	1	ed
2400	wistful	NONE	1	1	ly
2401	wrapp	NONE	1	1	ing

```
  2402    yield              NONE                        1             1   ing
  2403    zeis               NONE                        1             1   ing
  2404    zombi              NONE                        1             1   es




# Signature Count
# ---------------
  350

# Signature    Stem Count   Corpus Count
# --------------------------------------
# Remark
# ------
# Stems
# -----

  NULL.s   398   3518
  SF1
  achievement acre action administrator adult adviser aerial afternoon agent agreement
  allowance alternative amendment american ankle ant apartment appearance application appointment
  apprentice area argument arise arrangement assessment athletic attorney aunt average
  ball ballot banker bassi bat begin belief benefit bid billiken
  bird blow blue bridge brook brother builder building bundle burke
  burst butler camera camp candidate cardinal career catcher celebrate center
  chain champion championship chance charge choice clerk client cocktail college
  commitment committee communist compensation completion conservative consultation contractor con
  corp corporation correspondent course criminal cuban cut dancer daughter decade
  decide decision defendant delegation detective development dick dinner direction director
  disappointment disclosure discussion dissent district doctor dodger dog doing dollar
  door dot drawing duffer edward effort election employment error estimate
  event expense expert expire expressway eye fan farm father feat
  female figure fine fit folk food frame freeze fund fur
  get gift girl goal god golfer graduate group grover guest
  guy harvey hearing heart highway hill hit hitter holiday homer
  horse hotel hour human hundred hurler husband idea independent indictment
  individual influence inning inspection instance institution intention interview investigation i
  island issue item jail job junior justice kid knight kroger
  lao laotian law lawyer leaguer lefthander legislator legislature level liberal
  lie lighter loan longhorn lot loyalist machine maid major matter
  meeting member mile million minute misunderstanding model moral mortgage motel
  motion motorist movie mustang narcotic nerve neusteter neutralist newspaper obligation
```

observer obstacle offense officer official one operation opinion oppose oriole
our outlay pain paper path peddler pedestrian pension people performance
permit petition phone pirate pledge poll pop porter portion position
precinct prince princes privilege problem procedure professional professor profit program
project proposal quarterback queen race radio rain rate reactor rebel
recipient recommendation red reform relieve representation representative requirement response
retirement revenue revision rhode ribbon rifle road robert role rookie
rose rostagno run saving scholarship selection senator service session set
setback sheet shelter shooting shot shrine signal signature silver single
slipper slogan sometime son song source speaker spectator spirit spot
squad stand standard statement station stock stop store storm street
string stroke structure struggle student sub submarine suggestion supervisor surprise
sweet system talent task taste test texan thank theater thing
third thousand threat ticket touchdown tournament toward towel track traveler
tree trend trial tribunal trim trip triumph trooper trouble truck
truth twin type uncle uniform vehicle violation voter wacker wage
walter ward warden way wendell william writer yankee

’s.NULL 131 1873
SF1
adair administration alliance allison alusik anderson anne another army arnold
association atlanta authority baltimore barnard baylor berger berlin berry blanchard
body boston bride bridegroom britain brocklin broglio brooklyn brooks byrd
caldwell canada carreon castro chicago children christine city conference cotten
country county daniel danny denomination denver eisenhower else flock football
formby friday fuhrmann gannon gardner gee georgia gerosa gladden gordon
governor hall hansen house howsam hyde jenkins jersey kennedy kowalski
kunkel latter lonsdale lowe mantle marr maryland mc*connell meyner mickey
mills milwaukee mississippi molly monday navy nixon nobody nugent ordinary
organization palmer party phouma player portland railroad rayburn russell russia
ruth saturday secretary shaw she shea skipjack smith spahn stengel
stram sunday tech that throneberry thursday tomorrow tonight tuttle university
wagner wednesday weinstein wert wife willie woman women world yesterday
york

NULL.n’t 6 788
SF1
could did does had was would

’s.NULL.s 24 654
SF1
baseball chapter club commissioner controller court day department dresbach fall
force game geraghty leader mother other panel physician season shop

```
sister team union week

NULL.ly   58   526
SF1
absolute annual app apparent bad belated certain complete constant current
definite different entire equal exact former generous halting honest identical
immediate increasing intellectual like loose love main mental most narrow
over particular poor potential previous private prominent proper quick rapid
real recent repeated reported serious severe sharp short sore successful
sudden sure time ultimate unanimous unlike unusual usual

NULL.ing   41   467
SF1
approach beat boast border brief bring campaign carry clock combat
comfort debut deny draw fly fullback gather heat inn jockey
link load march picket rejoin respond sell send skylark staff
strengthen study supply switch thrill trust try undergo understand vacation
will

NULL.ed.s   40   397
SF1
account amount appeal arrest attempt back book bound check comment
concern condition defeat demand detail draft explain fear grant happen
insist intend mail mark mention merit plead rank repair result
seat seem sponsor stay subject succeed survey term want water



NULL.ed.ing.s   18   321
SF1
add ask assault attack award claim cover end help kick
look point question record talk total train word

e.ed.es.ing   14   316
Known_stems_to_suffixes
believ chang doubl emphasiz handl hous includ increas liv plac
promis provid receiv schedul

NULL.ing.s   19   279
SF1
bond boost break crowd feel guard keep know meet neighbor
plow request ring room say spend sport think throw


's.NULL.ly.s   3   256
```

```
SF1
month night year

e.ed   48   245
Check_sigs
advanc arous assur balanc celebrat challeng charg collaps combin damag
decid declin devot divorc estimat experienc expir fet forc hop
locat necessitat oppos pledg prais privileg prov recogniz releas reliev
reviv rout scrimmag shap singl slat slic solv squeez subdu
surpris telephon terminat tripl voic wag wav welcom

NULL.er   19   241
SF1
best bunt cent command fast few frank lay long must
nev old outfield palm prop roll roof tough young

ies.y   27   235
SF1
academ activit agenc authorit bod charit communit compan countr deput
dut famil lad majorit propert qualit safet societ suppl tall
territor testif traged universit utilit vacanc victor

NULL.ed   49   233
SF1
absorb acclaim accomplish affect avoid belt black block cloud coast
contact contend contest delay discredit display down earn enjoy furlough
infest jump knock land limit list mount murder mutter outclass
pardon protect push register rest restrain retain reveal romp rumor
smash smooth sound support suspect veil warm well widow

[snip...followed by these minimal cases at the very bottom:]


ations.ing.s   1   3
From_known_stem_and_suffix
confront

ations.er.ing   1   3
From_known_stem_and_suffix
observ

ary.ers   1   3
SF1
custom
```

```
  ern.ernal   1   2
  Check_sigs
  ext

  ers.ing   1   2
  SF1
  manufactur

  ent.ing   1   2
  SF1
  correspond

  NULL.ized   1   2
  SF1
  organ

  ation.ent   1   2
  SF1
  magnific

  ance.ing   1   2
  SF1
  disturb

  ment.ors   1   2
  SF1
  assess

  ies   1   1
  From_known_stem_and_suffix
  repl
```

0pt47K words

```
# Signature Count
# ---------------
  3139

# Signature    Stem Count   Corpus Count
# --------------------------------------
# Remark
# ------
```

```
# Stems
# -----


  NULL.s   2222   56559
  SF1
  abbreviation abernathy aberration abolitionist aborigine abortion absence absorption acceleratio
  accelerometer accolade accommodation accompaniment accompanist accomplice accomplishment accoun
  achievement acknowledgment acquisition acrobatic action adagio adaptation additive adherent adhe
  adirondack adjective adjunct adjustment administrator admission admonition advancement advertise
  aesthetic affiliation affirmation affliction afghan african afterward aggie agglutinin aggregati
  aggression agitator agreement ailment ainu airfield airplane airport airstrip alabama
  albanian alia alibi alignment alkali allegiance alley allocation allotment allowance
  alloy allusion almond alpert alsatian alteration amazon ambition ambulance amis
  amplifier amulet amusement anabaptist anachronism analogue ancestor andrena andrew anecdote
  anglo-*american anglo-*saxon anionic ankle announcement annoyance anode antagonism antagonist an
  anterior anthem anthropologist anti-*communist antibiotic anticipation antiquarian antique anywa
  apartment apostle appalachian appearance appetite appliance applicant application appointee appo
  apportionment appraisal appreciation apprehension appropriation approximation apron aptitude arc
  arena arhat arianist armament armchair armpit arrangement arrival arrowhead article
  articulation artisan arylesterase asian aspect aspencade aspirant aspiration ass'n assailant
  assemblage assertion assessment asset assignment assumption assurance athenian attachment attain
  attention attitude attraction audience authentication authorization auto automobile avenue aviat
  avocado axe axle aye azalea babe babylonian bachelor backbend background
  backward backyard badge bag balkan ballad ballard ballerina ballet ballistic
  ballot ballplayer banana bandit banister banshee bantu barbarian bard barnyard
  barrack barrel barricade barrier basement basic basket bassi bathroom bathtub
  bathyran battalion batten battlefield bauble bawh bayonet bazaar beadle beaker
  bean bearden bearing beating beatnik beep beer begin beginning behold
  belgian belief bellboy belonging bemoan bentley bequest bereavement beside bespeak
  beverage bicep bicycle bidder bifocal billboard billet billiken billing billion
  binder biographer biologist biscuit blade bleeding blessing blizzard bloke blouse
  blower blueprint boasting boatel boatload bodybuilder boite bombing bonfire bookcase
  booking booklet boomerang booth bootlegger borden borough bosom bostonian bottleneck
  bough boulder boulevard bouquet bourbon bovine bowl boxcar bracket brake
  breakdown breaker breakthrough breakup breakwater breeze brigade briton broadcasting brochure
  bronc bronchiole brothel bucket buddhist buena buffoon bulkhead bull's-eye bum
  bumblebee bunkmate bunter bureau burlesque burning bushel butler byproduct cabana
  cabinet cadillac cafe cafeteria calculation calendar caliber calibration caliper camel
  cameo campground canal cancer candidate canister canoe canyon capacitor capsule
  captive carbine cardinal cares carriage carrot carryover cartoonist cartridge carving
  cask castle castorbean catalyst caterpillar cathedral catkin catskill ceiling celebration
  cellar cellulose centimeter ceramic cereal cetera chairmanship chambermaid championship chandeli
  chapel chaplain characterization charting chartist cheek cheekbone cherokee chestnut cheyenne
  chicken chiefdom chieftain chimney chip chive chloride choctaw chord chowder
```

christopher chromatic cigarette cinder cipher circonscription circumstance citation civilian cl
clap claret classification classmate classroom cleft cliche cliff climate clip
clique clod closeup clothesline clue clump coating cobblestone cockpit cocktail
coconut coed coefficient coincidence coke collaborator colleague collection collision colman
colored columnist combatant combination combine comedian comic coming commencement commentator
commitment commonplace commonwealth commune communist comparison compartment compatriot compel
competitor compilation complaint completion complication component composite compulsion compute
conception concessionaire conclude conclusion concur confabulation confederation confessional c
confinement conformist confrontation confusion congratulation conjunction connection connoisseu
conquest conscience consequence conservative consideration constantino constituent constriction
consultation contention context contingent contraceptive contradiction contribution control con
conviction convocation cookie cooperative coping corduroy core corinthian correction correlatio
correspondent corridor cosmetic cosmo cossack cottage counselor counterpart coupon courtier
courtyard covenant covering cowbird coyote crackpot cramp creation creator creature
creek creeper crevice crib cricket criticism critter crop crossing crystal
crystallite cuban cubist cuff culprit culver cupboard curd currant curriculum
curry curtis cutter cutting deacon deadline deadlines dealing debt decide
decimal declaration decoration decorator deductible deduction deed deferent deferment deficit
definition degree dejeuner delaware delegation deliberation delimit delinquent delta deltoid
demonstration denial denunciation departure dependent deposition depot depression deprivation d
derivation descendant description designation desk desolation dessert detector detergent determ
determination detractor deviation device diagram dialogue diameter diamond diehard difference
dilemma dinosaur dip dipole directive disadvantage disagreement disappointment disaster disburs
disc discipline disclosure discussion dislocation disposition disruption dissatisfaction dissen
distance distinction distortion distraction distribution district disturbance ditmar divan divi
doctrine doe doing dolphin domain donation donor dooley doorway dosage
douglas dozen draftee dragon drama dramatic dramatist drawer drawing dressing
drier drinker drip driveway drone drop dropping drought drugstore drum
dud duet duffer dumbbell dupont duration dweller dwelling earning earthquake
easement eatable eating eccentric echelon ecumenist edition effluent egyptian election
electroshock elegance elimination elizabethan elk ellipsoid eluate emanation embodiment emerald
empire employment encyclopedia endearment endeavour ending endowment engagement englander engra
enlargement enrollment ensemble entail entertainment enthusiasm entrepreneur epidemic episode e
eqn equation equilibrium equine equivalent error escapade escutcheon eskimo essence
establishment estate esthetic ether ethicist evade evaluation evasion evil evocation
ex-*president exacerbation exaggeration exaltation examination example excavation excel excelle
exclamation exclude exclusion excursion executor exemption exertion exhibition exit expectation
expedition expenditure expense experimentation explanation explode exploration exposition expos
extension exterior extractor extrapolation eyeball eyebrow eyelid facet faction failure
fairway falcon falsehood farce farmhouse farmland farnese farrell fascist fastening
fathom favorite feat fed feeding feeling fella fellowship femme fender
fermentation fern ferraro fertilizer fervor festival fiat fiber fighter filament
filbert filibuster filipino filling finalist finder finding fingering firecracker fireplace
fitting five fixture flag flake flannagan flannel flavoring flea fledgling
fleming flight flip floe flop floridian flotilla flyer foal foe

food footfall foothill footnote footstep foray ford forearm forefinger forehead
foreigner forerunner format formation formulation forum fosterite fragrance franchise franciscan
freeholder freeway freighter french-*canadian friar friendship frieze frog frolic frontier
frustration fugitive funeral furnishing galley gallstone gambit gangster gardenia garment
gassing gateway gathering gaucherie gazette gelding gender generalist generalization generator
genre gentian gentile geologist georgian gershwin get ghazal ghetto ghoul
giant gibe gingham giveaway glacier globe globulin glycol goal going
goitrogen golfer grab gradient gram grandfather grandson grape grapevine gras
grassland graveyard graybeard greek greenhouse greeting grenade grievance grinding grouping
grower growth grub guarantee guerrilla guest guise gunner gym gymnastic
gynecologist gyration gyro haircut halfback halfway hallelujah hallmark hallway halo
ham hamburger hamiltonian handbook handful handgun handicap handkerchief handstand hangar
hangover happening harding hardship harvey haven haystack haze headache heading
headland headquarter headstand hearing hebrew heinze helmet hemorrhage herb heretic
heritage heroic heron herpetologist hessian highland his hitter holding holdup
holiday hollyhock homecoming homemaker homeward homosexual honeybee hoodlum hoof hookup
hoosegow horizon hormone horror hose hostage hound householder hub hugging
huntington hurray hutment hydride hydrocarbon hydrogen hymen hymn idea identification
illumination illustration illustrator imagination imagine imagining imbalance imitation immigran
implication improvement improvisation impulse inboard incentive incitement inclination include i
incompatible incompetent incumbent indication indicator indictment indoor inducement induction i
inference infestation inflection informant infringement ingredient inhibition injunction injusti
inlet inmate inning innovation inoculation inscription insect insecticide insertion inset
insight insignificance insinuation inspection installation installment instance insulator insurg
intangible integer integral intendant intensifier interaction interface interferometer interior
intermediate intermission internationalist interpenetrate interpolation interpretation interrela
investigator investment investor involution involvement irritation isle israelite italian item
jab jake jar jaw jaycee jean jeffersonian jerebohm jerking jeroboam
jesuit jowl judgement judgment judson juice julep juncture jungle jurist
juror justification katangan kenning kernel keynote kidney kilometer kilowatt kingdom
knee knob kochanek korean kraut laban lagoon lamb lamechian lamentation
landing landmark landslide lane language lantern lao laotian lap lapel
las lashing latitude launching laundering laurel lawn lawsuit leading leaflet
leaving lefthander legion legislator lemma lemon lesson lexicostatistic liaison libertarian
libertine lien lifeboat ligand lilac limitation limousine lindemann lineage liniment
liquidation listing literature lithograph litigant loading lobule location locomotive lodging
logarithm loin longhorn longing longitude loophole lotion loudspeaker loyalist lui
luncheon machinist magistrate magnate magnetism magnitude magnum magpie maguire maiden
mailing maitre makeshift making maladjustment mamma mana manifestation manikin manipulation
mannerism manor manuscript maple marble mardi marketing marking marriage martian
martini masterpiece matisse maverick maximum meadow measurement meat mechanism medal
medici medicine meditation meeting megaton membership memoir menarche mennonite menu
merchant message messenger metabolite metal metaphysical methuselah mexican meyer micelle
micrometer microorganism microwave midst mig migrant milestone millidegree milligram millionaire
milquetoast miniature mink minor minstrel miracle miscalculation misconception misconstruction m

```
misfortune misrepresentation mission misunderstanding mixture mme moccasin modification modifie
molecule monkey monograph monomer monosyllable monster mop morphophonemic morsel mortal
mortgage mosaic moslem mosque motel motet motif motivation motorist moulton
mountainside mounting mouthpiece movement movie mug mule multitude museum musical
musing musket muslim mustang muzzle mysticism nap napkin narcotic narrative
nationalism native navel navigator necklace negociant negotiation neighborhood nephew neusteter
newcomer newlywed newsletter newt nickel niece nightingale nikolai nip nitrate
nomia non-*catholic nonconformist normal northerner nostril notebook notion noun nozzle
nuance nude nuf nuisance numeral nut nutrient nymph nymphomaniac o'*dwyer
obedience objector observance obsession obstacle occupant occur occurrence octave odor
offense offering oilseed olympic omission onion onlooker onset onslaught onward
opening operand operator oracle orange oration orchard orchestration ordering ordinance
ore orgasm orientation oriole orthodontic orthophosphate ounce our outboard outbreak
outburst outcast outcome outdoor outfielder outlay outlet outpost outrigger oval
oven overall overcoat overhang overlap overture owen ownership oxygen oyster
packard packet pad paean pagoda pail painting pajama pakistani palazzo
pamphlet panorama panther paperback parable parachute parade parameter parapet parasol
parlor participant particle parting partisan partition passenger pastel pastime pasture
patina pavement pavilion payment peacock peanut pebble pecan pedal pedestrian
peg pegboard pennant peptide percentage perception performance persian personage perspective
persuasion perturbation phase pheasant phi philippine phillip philosopher phonemic phonetic
phonograph phosphate photo photocathode phrasing physicist piano piazza pickoff picnic
pigeon pilgrimage pillow pinnacle pinning pirate pistol piston pitfall plaid
planetoid plantation planting plaque platform platoon platter playback playhouse playmate
playwright plaza pleasure plug pocketbook poem poetic politician politico polybutene
polyester polyether polyisocyanate polymerization polynomial polyphosphate populaire population
positivist possession postcard posture potboiler poultice practitioner pram prank preamble
precedent precept precinct predecessor prediction predisposition prefecture preference prelude
premium premonition preoccupation preparation prerogative prescription presence pressure presum
pretense pretext prevision primate princes principle prisoner probing procedure proceeding
processor proclamation proctor production profile progression projectile projection prolusion p
pronouncement proponent proposal proprietorship propulsion prosecutor prostitute protease prote
psychiatrist psychologist pub publication puddle pulley pulling pulpit pulsation punishment
punk pup pupil pursuit purveyor put pyrometer qualification quarterback questionnaire
quintet quota quotation rabbit racketeer radiation radiator railway rambling ramification
rapture rascal rathbone rating ratio rationalization rattlesnake ravine ray reaction
reactor reading reagent realm reappraisal reb rebel rebellion receipt recherche
recipe recipient recital reckoning recognize recollection recommendation recording recriminatio
reduction redwood reef reference referral refinement reflection reflector refreshment refrigera
registration regret regulation rehabilitation rehearsal reimbursement reinforcement rejection r
reminiscence remnant rendering rendition rental renunciation reorganization repetition replacem
repression reprisal reproduction repulsion requirement reservation reservoir residence residue
resistance resistor resolution resonance response restaurant restriction resultant retailer ret
reunion revelation revelling reverberation rheumatic rhode ribbon rican ringing rite
riverbank roadway robertson rodent rodeo roger rogue role rolls-*royce rooftop
```

rookie roommate rostagno roundup ruffian rumanian runner runway rupee rutabaga
sabina sable safeguard sailboat salad salon saloon salvo samuel sap
sarcasm satellite satisfaction sausage sauterne saving saying scaffolding scandal scandinavian
scapegoat scenario scenic scholarship schoolboy schoolmate schweitzer scientist scimitar scion
scoreboard scoundrel scraping screening scripture searchlight secant secessionist secretion sec
sector sedan seedcoat seeker seismograph selection senior sentinel separation sergeant
sermon servant serving session setback setting settlement shading shaft shareholder
shaving shawl shibboleth shim shipmate shipment shirt shoelace shoestring shooter
shooting shore shoreline shortage shortcut shot shotgun shred shrine shrub
shrug shun shut shutdown sideboard sideline sidewalk siecle sierra signature
signpost silicate silo siren sitting situation skeleton skid skip skit
skull skylight skyscraper slap sleeve slicker slip slitter slogan slug
slum smelt snack snag snowball sociologist socket soiree sojourner solitude
soloist solution solvent sometime somewhere sonata sonnet soothsayer sop sophomore
soprano sorrow source souvenir soybean spacesuit spacing spade spasm specialist
specie specification specimen speck spectacle spectator specter speculation speculator spewing
spire spotlight spouse sputnik squadron squall staccato staircase stairway stalling
stance standard statement statute steak steelmaker steeple steiner stem stepmother
steroid stetson stimulant stimulation stir stockholder stocking stomach stop stopover
stoppage storehouse storyline stove strait strap stratagem straw streetcar striving
stubblefield studio stunt subdivision subjectivist submission subpoena subroutine subsection su
substance substrate subsystem subtype suburbanite subversive subway suffering suggestion suicid
suitcase suitor super-*set superlative supermarket superstition supper supplier surcliffe surfa
surgeon surrealist surrounding survivalist survivor suspension suspicion sweeney sweetheart swel
swivel syllable synagogue synthetic syrian szold tabernacle tablespoonful tablet taboo
tabulation tackle tag takeoff taking tango tanker tantrum tappet tarpaulin
task tavern teaching teahouse teamster teaspoonful technician technique teen tektite
telegram teletype teller temperature tempo temptation tenement tenor tentacle tenth
terminal terrain terrier testament testimonial testing texan textbook thaxter theater
theatergoer theologian thermocouple thermometer these thicket thigh thing thoroughfare thousand
throne thruway thug thynne ticket tidbit tide timetable tip tissue
titer titter toe toilet tombstone tong topcoat topping torrent torso
tortoise touchdown touchstone tournament township tracing tractor trademark trance tranquilizer
transaction transducer transient transistor translation transmit transom trap trapdoor trapping
travelogue tray treatment trend trestle trial triangle tribunal tribute trim
trimming trinitarian trinket triplet tripod trooper troopship trough trouser trunk
tulip tumor turbine turkey turning turnout turnpike turret turtle twinge
typewriter tyrant ukrainian umbrella undergraduate underlie undertaking unification unknown upl
uprising upshot upward urethane urging urn usage user utterance vagabond
valuation valve variable variation vase vector vegetable vehicle velour vendor
ventricle veranda verge version vertebrate vessel veterinarian viewpoint village villager
vine vineyard violation violet violinist virtue vision visitation visitor vista
visualize vitamin void volcano voltage volume vowel wacker wagon wallpaper
walnut wandering warden warrior wart wary watching watercolorist waterfall watershed
waterway wavelength wedding weekend wendell westerner westward wherefore whig whim

```
whispering whore wicket winning withstand wohaw woodward wop workout workshop
wrap wrestling wring wrist writing yachtel yankee yarn yeast yokel
your zombie

NULL.y   62   29522 ** LOTS OF FUNNY ONES
SF1
abbe alla and astra ** bets blake ** buckle burl ** carne ** carve
conciliator cone connall connell copper crank creamer dever dicke dirt
dishonest donna donnell dusk filth flesh fluff fog grubb handle
hire immodest joss kentuck loft lund lura orthodox pals paunch
photomicrograph pith pose potter prior quyne rall rand regulator scrutin
slipper soma spider stale swank syrup tartar teens thrift tips
tweed velvet

's.NULL   924   20424
SF1
a*a*u abbas acheson adair adams adenauer admassy administration aeschbacher agamemnon
agriculture ahmad ailey aircraft alex alexander alix allen allison alusik
amadee anderson andrei andy angelo anniston announcer another anthony antoine
anyone apollo arbuckle aristotle arlene armory army arnold artery arthur
askington athlete atlanta attacker auctioneer augusta augustine austin authority b'dikkat
b*b*c balaguer bancroft bang-*jensen banks barber barco bari barnard barton
basil batista baylor beauty beckett beebe beethoven beige benefactor benson
beowulf berger berman bernini berra berry betty blanchard blanche blatz
blonde bob bobbie body bolingbroke bomber bondsman boniface bootle borromini
brace bradley brandon brandt brannon brenner bridegroom bridget britain broadway
brocklin broglio brooklyn brooks bruckner brumidi brush-off bryan buell bultmann
burch burlington burnham burns burton byrd cabot caldwell calhoun california
caltech cambodia canada cane capone cappy carla carmer carreon carwood
casey casino catcher catherine cathy celie chabrier chambre chandler channing
charley charlie charlotte chicago childhood children chiropractor choir chopin christine
chronicle chrysler cicero cimabue city clarke claude clayton clergyman cobb
colcord coleridge colmer colony colorado comedie commissioner communism community company
composer conant concetta congo congressman connecticut conrad consumer controller coroner
costaggini cotten cotter cotton coughlin county couple craig creston crombie
crosson cuba culture cunard cunningham curzon custer czarina daddy dade
dalton dana dandy danny dante darling dartmouth dave davidson davy
de*kalb deegan delphine denny denver deputy designer detroit devey diane
dickey dictionary digby diman django doaty dodge doolin doolittle dostoevsky
doyle drexel driver dronk drummer dufresne dulles dwyer earthmen edison
edythe egotist eichmann eileen eisenhower ekstrohm elaine elec else emerson
emile emma emmett employee en-lai enemy enright erikson ernie estella
eugene everybody everyone everything executioner faber faget family farmer favre
feathertop february felice felix ferguson fiedler fielder fink finney fisherman
flautist fleisher florida florist flotte floyd flynn forbes foreman formby
```

fosdick fox france francesca francie francisco franklin frayne fred freddy
freeman frelinghuysen frenchman fritzie fromm fudo fuhrmann gallery galtier game
gannett gannon gardner gargery garibaldi garth gavin gaylor geely georgetown
georgia germany gerosa getz giffen gilborn gladden gladdy glendora glimco
globocnik goat goethe gogol gordon gore gorham gosson grabski gracie
grafin grandma granite granny grant greece gregory greville griffin griffith
grigori groth guardino gulf guthrie hale hammarskjold hammett hampton hamrick
haney hangman hansen harburg hardy harlem harmony harriet harrington harris
harrison harry harvard haumd hausman havisham haydn hearst hector heidegger
heidenstam helion helva hemingway hemisphere henrietta henry herberet herford herman
herry hetman hetty hillman hilprecht hino hirey hirsch hogan hoijer
holden hollywood horace horne houghton housman howard howe howsam hrothgar
huckster hudson hugo hume hunter huxley hyde ike india indiana
industry ingleside inspector institute israel istiqlal italy izaak jackie jacoby
jane jannequin january jed jehovah jenkins jennie jenny jerry jersey
jessica jesus jeweler jim joan joel johnnie johnson jonathan journal-*bulletin
juanita jubal juet kahn kai-shek karipo kate katharine katherine katie
kayabashi keith kemble kennan kennedy keys killpath kipling kirby kirov
kitti kitty knowlton kornbluth koussevitzky kowalski kremlin kruger krutch kunkel
la*guardia larson latter lattimer lauchli lawman layman laymen leavitt leesona
lenin leningrad lenygon leonato lester letch lewis liberty lillian lilly
lincoln linda listener littlepage lizzie lloyd lockheed loesser lolotte longfellow
longshoremen lonsdale lovejoy lowe lowell lublin lucas lucifer lucille lucy
luke lumumba lyford lyricist mac*donald macaulay mack macklin madison madonna
mae maestro maggie magwitch mahler mahzeer maitland majesty mallory malraux
mama manchester manhattan mankind manley manufacturer marcel mare maris marlin
marlowe martha masu matsuo maude maxine maxwell mc*carthy mc*clellan mc*cloy
mc*cone mc*connell mc*kinley mc*pherson means medfield meeker meltzer mendelssohn mercer
meredith mexico meynell meyner miami michelangelo mickey mid- mijbil milhaud
militarist millay miller mills milwaukee minnesota miranda miriam mississippi missouri
mitchell moliere molly mommy mongolia monmouth montero montgomery moore morgan
morgenthau moriarty morse morton moscow mossberg mozart mulligan mundt munich
municipality murderer murphy murray musmanno mussorgsky myra n*c nadine nagrin
nassau nasser nate navy needham nehru newbiggin newport nicolas nixon
nugent o'*banion o'*connor o'*donnell oats observer oersted oldenburg oliver olson
ontario ordinary orlick ortega oso othon oxen oxford pagnol painter
palfrey palmer pam pamela pandora pantheon papa parker parry partlow
party pasadena patchen patrick patrolman patronne paula pauling pawtucket peabody
peale pedersen pendleton penny pentagon perier perrin petitioner pettigrew philadelphia
philip phouma picasso piepsam pietro pike pilate pimen player poetry
poitrine poland policeman policemen pollock pompeii pont pony pope pops
porter portland potemkin powell printer prokofieff providence ptolemy pumblechook quake
quasimodo quebec quiney rabbi rachel racine rameau ramey rangoni rankin
rayburn reavey receptionist rector remarque rembrandt rev rexroth rheinholdt richardson
rider rifleman riflemen ritter riverside robby romeo roofer roulette rourke

```
ruger runyon rusk russell salter sanctuary sangallo santa santayana sarah
satan saud saxton schiele schonberg schoolmaster schopenhauer schubert schuman schuyler
scotty seaton secretary seebohm segovia seller semester senate sentry seward
sewer shaefer shafer sharpe shaw shayne shearing shelley sherman shirley
sibylla sidney sihanouk simon simpson singer skipjack skolman skolovsky slater
slocum slope sniper snyder society solomon somebody someone sorrentine sparling
sportsmen sprague springfield stallion stanley steele stengel stephens stevenson stewart
stone storyteller stram stranger stravinsky strindberg sturley suite sukarno sulky
sulzberger susan susie suvorov swadesh sweden symphony syndicate t*r tahse
tailin teacher tech telegrapher tennessee thayer thelma theology thet thomas
thompson thoreau throneberry thurber thursday tilghman tillie today todman tolley
tolstoy tommy tomorrow tonight toscanini trafton trapper treasury trevelyan tribune
truman tucker tuesday tuttle twain u*n u*s u*s*s*r udall undersecretary
underwood university uno varlaam vec*trol verloop vermont vernon victoria vidal
vienna virginia vivian voltaire wagner wally walsh warsaw washington washizu
watson watson-*watt weinstein welch wert wesker wheeler wheelock whipple whirlwind
whitehead whitman whittier williams willie winslow wisconsin wisman wolfe wolff
wolpe women woodbury woodcock woodruff woods worker wright writer wycoff
xavier yale yesterday zachrisson

NULL.ly   607   15941
SF1
absurd abundant accidental according accurate accusing acoustical active actual actuarial
acute adamant additional adequate administrative admiring admitted advantageous adverse advised
affecting affectionate affirmative agile agricultural aimless alarming alleged alternate amazin
ambitious amorphous amused amusing analogous analytical anhydrous annual anxious appalling
apparent appraising appreciative approving approximate arrogant assured astonishing astronomica
attractive aural auspicious austere authoritative axial bare beautiful behavioral belated
belligerent bewildered biblical biological bleak blissful blithe breathless brilliant broken
categorical causal cautious ceaseless ceremonial charming chronological classical clinical coll
comparative compassionate competent competitive comprehensive conceded conclusive concrete conc
consanguineous considerate consistent conspicuous constructive consummate contemptuous contente
continuous convenient converse convincing convulsive copious corresponding cortical courageous
covert crucial cultural curious cynical dangerous decent deceptive decided defiant
definite deliberate delicate delicious delightful demanding denominational depressing despairin
determined devastating devoted devout dialectical diffuse diligent dimensional disconcerting di
dismal dispassionate disproportionate distal distant distasteful distinctive distracted disturb
doctrinal dogged dominant doubtful doubting dour dramatical dreadful dreamless dynamical
economical efficacious efficient effortless elaborate electrical elegant eloquent embarrassing
emotional empirical enchanting encouraging endless enduring engaging enormous enterprising envi
epicyclical equidistant erroneous ethical eventual everlasting exasperating exceeding excellent
excessive excited exhausting exhaustive expectant expected expeditious experiential extensive e
exuberant facetious faithful fascinating fearful fearless ferocious fervent feverish financial
fitful flagrant flamboyant flattering fluent focal former fortunate frenzied frightening
frightful frowning fruitless furtive gasping generous gentleman genuine geographical geometrica
```

```
ginger girlish glaring gleeful glib global glorious glum gorgeous governmental
graceful gracious grammatical graphical grateful gratifying gratuitous habitual haggard halting
haphazard harmless harmonious heated hesitant hesitating hideous hilarious historical homogeneou
horizontal horrifying humane humiliating hurried identical illegal imaginative immediate immense
impassive impatient imperious implicit important imprecise improper impudent inadequate inadver
incessant incoherent inconspicuous inconvenient increasing indignant indolent industrious infini
infrequent ingenious inherent insane insidious insolent instantaneous instinctive insufficient i
intense intensive intentional interesting intermittent intimate intricate intriguing intuitive i
ironical isothermal jagged jocular joyful joyous jubilant judicious knowing laborious
laughing legitimate leisure lewd linear listless logical loving magical magnificent
malicious marked marvelous masterful mathematical mc*fee mechanical mental merciful merciless
methodical meticulous metrical microscopical militant minimal miraculous mistaken mocking modera
morose most mountainous mournful moving mute mysterious nearsighted needless negative
nominal noncommittal nondescript notorious numbing numerical oblique obscure occasional ominous
ontological operational optical oral organizational ornate outstanding outward overwhelming pain
painless painstaking paradoxical partial passionate peaceful peculiar perennial perilous periphe
permanent perpendicular perpetual persistent persuasive pervasive perverse philosophical physiol
pious pitiful pitiless poignant pointed political positive prayerful precarious precise
precocious predominant preferential premature previous private professed profuse progressive pro
proportionate protective provocative prudential psychical psychological pungent purported purpos
qualitative quantitative quarter querulous questioning racial rakish random raucous reassuring
rebellious recent recurrent refreshing reluctant repeated reported reputed residential resigned
resolute respectful respective restive reverent rhythmical ridiculous rightful rigorous rollicki
rugged satirical scarce scathing scornful scrupulous seasonal secure sedate seeking
seeming selective senseless serene severe shattering shining shocking shy significant
silent similar simultaneous skeptical skilful sleepless sluggish smiling smoldering snobbish
snug sobbing sociological sodden sole soothing soulful sparse spectacular spectral
speculative spontaneous staggering stark startling statistical stolid strenuous striking structu
studious stunning subconscious subjective subsequent substantial substantive successful successi
sufficient sullen superb supine supposed supreme sure surprising surreptitious suspicious
symbolical symmetrical syntactical tactical tactual tantalizing taunting technological tedious t
temporal tempting tenacious tense tentative tenuous terse theoretical thermal thoughtless
threatening tireless transverse tremendous triumphant trusting twirling ultimate unambiguous una
unceasing uncommon unconcerned unconditional unconscious uncritical unerring unexpected unfailin
unhurried unilateral uninterrupted unknowing unlike unobtrusive unofficial unqualified unrestric
unsmiling unsuccessful unusual unwise unwitting urgent usual vain valiant various
vehement vertical victorious vigorous violent virtual vocational waspish willful wise
wistful woeful wondering wondrous worried wry zealous

's.NULL.s   199   13466
SF1
actor adolescent afternoon agent airline albright alliance ambassador amendment anaconda
analyst animal area assessor association attorney baseball bastard bedroom beginner
benet blackwell braque bride browning buckskin builder building burke burman
burnside buyer canadian captain caravan carolina carpenter cezanne chapter client
```

```
  club collector college colonel communicator concerto conductor conference constable contractor
  coolidge corporation cousin cowboy crosby customer cylinder daniel daughter daylight
  dealer defendant demon detective dreiser dresbach drunkard duke eagle economist
  eddie educator emperor era evening executive fan female football foundation
  frankfurter fraud freedom friday furnace gasket generation geraghty goulding gourmet
  governor grandmother guy harper historian hitler hood injun instructor ireland
  janitor junior justice kaiser khrushchev kid krim kroger lalaurie landlord
  larkin lawyer legation legislature let liar librarian lieutenant magazine magician
  management mansion mechanic missile mob monday morning mussolini nature network
  nightclub novelist opponent orchestra orthodontist owl palace payne physician plaintiff
  podger postmaster poussin pride prosecution pullman puppet queen raphael reader
  realtor registrant reputation respondent river rooster ruling sandburg saturday science
  sculptor shepherd sheriff shop sister sitter skiff sleeper snail sophia
  southerner soviet squire student styrene sunday superintendent target taxpayer taylor
  teammate textile their therapist tiger tourist transferor tribe trujillo union
  valley veteran walter way wednesday whip william yorker youngster

  NULL.ed.ing.s   174   11182
  SF1
  abound add administer affirm afford amount appeal arrest assault attempt
  audit await awaken award beckon belong belt bevel blast blend
  boast bolt border broaden burrow cancel carpet claw click climb
  cluck cluster coil compound concern contact contrast crawl creak crown
  curl decay deck discount display drill drown duck endeavor eschew
  escort exceed exclaim explain extend fasten filter finger flavor flounder
  frown gap gasp gather glow groom happen harrow haunt hoot
  hover howl insult interest kick kneel knock lack lean leap
  lessen litter loom loosen lurk maintain mention model mold monitor
  mortar mount nail neglect number obey overlook patent peel pertain
  picket pour proceed proclaim pump purport rasp recall reckon recount
  reel regain register reign remain remember represent resort retreat return
  reveal revolt reward roar scatter scream seem sheet shield shoulder
  shout signal skirt slant smell sneer spell spray squeal stack
  stamp stay steer straighten strand strengthen succeed suspect sustain swallow
  swarm swell swoop taunt taxi thread threaten thumb tilt trust
  unload unlock veer veil vein volunteer vow wail want weaken
  whisper wound yelp yield

[big snip]

# Stem Count
# ------------
  17260

# Index | Stem                  | Confidence           | Corpus Count | Affix Count | Affixes
```

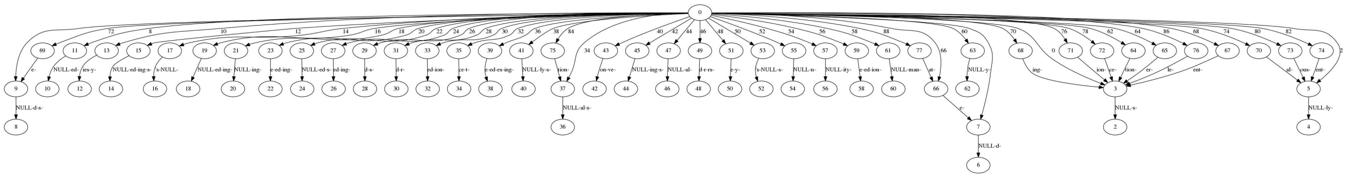
```
# ------------------------------------------------------------------------------------------
 1   **  the          From_sigs_find_stems     75026      13   ’s NULL a e
 2       and          From_sigs_find_stems     28856       2   NULL y
 3       that         From_sigs_find_stems     10779       3   ’d ’s NULL
 4   **  was          From_sigs_find_stems      9852       3   NULL son te
 5   **  for          From_sigs_find_stems      9718       9   NULL d e est
 6   **  with         Check_sigs                7646       7   NULL al er e
 7   **  his          From_sigs_find_stems      6994       2   NULL s
 8   **  not          From_sigs_find_stems      4963       8   NULL ation e
 9   **  are          From_sigs_find_stems      4718       3   NULL a s
10   **  but          From_sigs_find_stems      4391       2   NULL ton
11   ??  you          From_sigs_find_stems      4329       5   ’d ’s NULL r
12       hav          From_sigs_find_stems      4233       3   e en ing
13   **  her          From_sigs_find_stems      3904      11   NULL d e eti
14       one          From_sigs_find_stems      3476       5   ’s NULL ness
15   **  all          From_sigs_find_stems      3095       8   NULL a an en
16   ** she         From_sigs_find_stems     3060       8   ’d ’s NULL a
17       c            Check_sigs                3044      20   NULL a age a
18       there        From_sigs_find_stems      2851       5   ’d ’s NULL i
19       their        From_sigs_find_stems      2689       3   ’s NULL s
20       thei         From_sigs_find_stems      2668       2   NULL r
21       who          From_sigs_find_stems      2591       6   ’d ’s NULL a
22       bee          From_sigs_find_stems      2535       7   ’s NULL hive
23       has          From_sigs_find_stems      2447       3   NULL te ty
24       man          From_sigs_find_stems      2405      18   ’s NULL a do
25       mor          From_sigs_find_stems      2372       4   NULL al e to
26       will         From_sigs_find_stems      2333       7   NULL a ed fu
27       more         From_sigs_find_stems      2245       4   ’s NULL land
28       out          From_sigs_find_stems      2143       7   NULL do er f
29       other        From_sigs_find_stems      2119       4   ’s NULL s wi
30       eve          From_sigs_find_stems      2104       5   NULL n nt r
31       what         From_sigs_find_stems      1963       4   ’d ’s NULL m
32       tim          From_sigs_find_stems      1953      11   ’s NULL e ed
33       them         From_sigs_find_stems      1853       5   ’s NULL atic
34       new          From_sigs_find_stems      1844       9   NULL er est
35       can          From_sigs_find_stems      1797       6   NULL al e in
36       oth          NONE                      1708       2   er on
37       year         From_sigs_find_stems      1690       7   ’s NULL book
38       some         From_sigs_find_stems      1641       3   NULL day tim
39       som          From_sigs_find_stems      1628       4   a atic e ers
40       fir          From_sigs_find_stems      1618       5   NULL e ed in
41       time         From_sigs_find_stems      1610       5   ’s NULL less
42       state        From_sigs_find_stems      1596       7   ’s NULL less
43       these        From_sigs_find_stems      1574       2   NULL s
44       like         From_sigs_find_stems      1484       6   NULL e ly ne
```

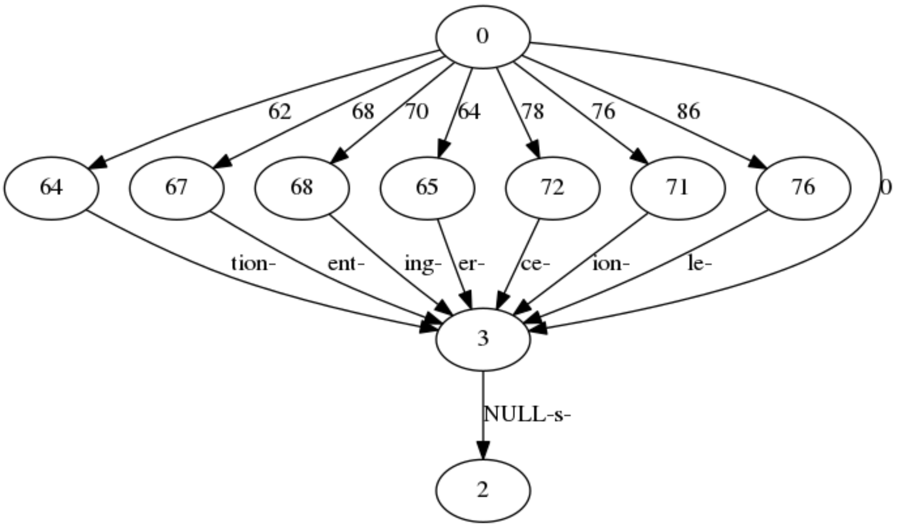
```
  45      may          From_sigs_find_stems     1422       6   NULL e er o
  46      two          From_sigs_find_stems     1415       3   NULL s some
  47      any          From_sigs_find_stems     1386       4   NULL e time
  48      first        SF_1                     1362       2   NULL hand
  49      lik          From_sigs_find_stems     1361       3   e ed ing
  50      use          From_sigs_find_stems     1338       7   NULL able d
  51      work         From_sigs_find_stems     1311      13   's NULL able
  52      even         From_sigs_find_stems     1306       3   NULL ing ly
  53      see          Check_sigs               1284       7   NULL d in in
  54      too          From_sigs_find_stems     1280       3   NULL k th
  55      our          From_sigs_find_stems     1279       2   NULL s
  56      over         From_sigs_find_stems     1269       7   NULL age han

[huge snip]

  17237   wust         NONE                        1       1   man
  17238   wym          NONE                        1       1   an
  17239   wynst        NONE                        1       1   on
  17240   xavi         NONE                        1       1   er
  17241   xen          NONE                        1       1   on
  17242   xylophon     NONE                        1       1   es
  17243   yali         NONE                        1       1   es
  17244   yapp         NONE                        1       1   ing
  17245   yardum       NONE                        1       1   ian
  17246   yedis        NONE                        1       1   an
  17247   ying         NONE                        1       1   er
  17248   yond         NONE                        1       1   er
  17249   yong         NONE                        1       1   st
  17250   yonk         NONE                        1       1   ers
  17251   yoshimoto    NONE                        1       1   's
  17252   yucat        NONE                        1       1   an
  17253   zachriss     NONE                        1       1   on
  17254   zamiatin     NONE                        1       1   's
  17255   zaporog      NONE                        1       1   ian
  17256   zaroub       NONE                        1       1   in
  17257   zeitge       NONE                        1       1   ist
  17258   zenn         NONE                        1       1   ist
  17259   zin          NONE                        1       1   man
  17260   zomb         NONE                        1       1   ie
```

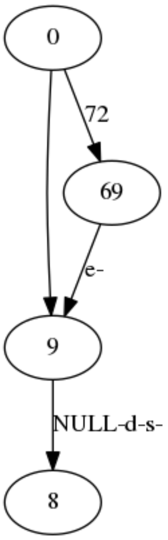
Linguistica 4 outputs a log file with a great deal of information in html.

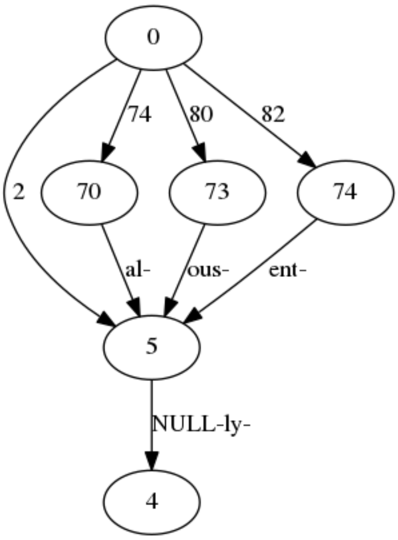
5.4.3 Linguistica 5

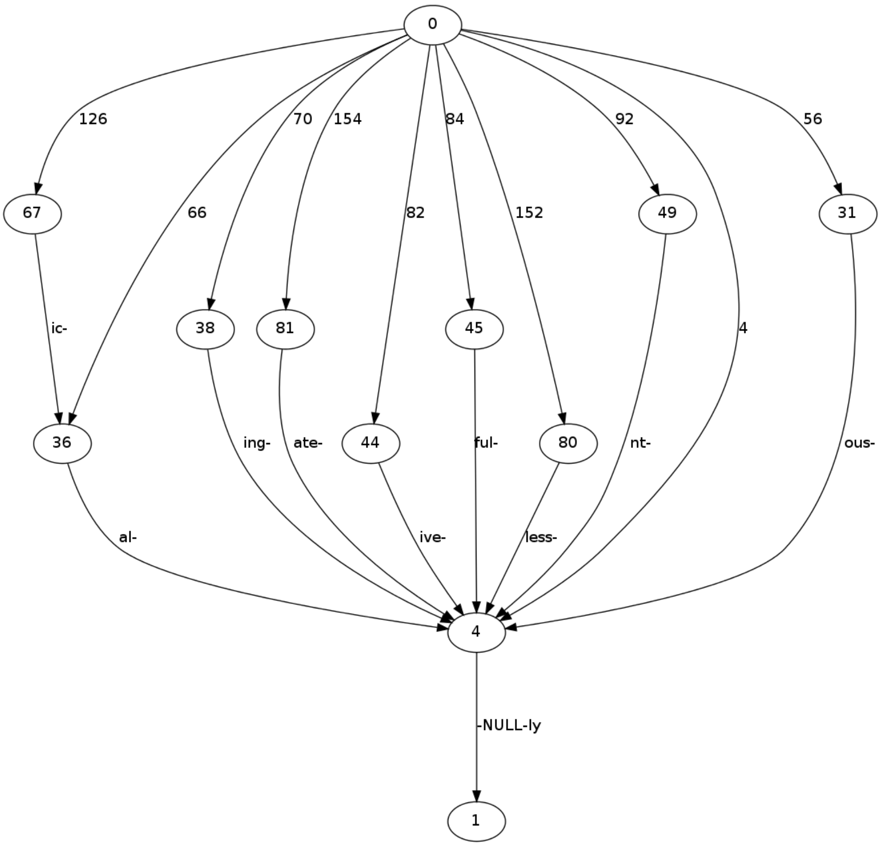
Linguistica 5 begins by making signatures, but with much more liberty than in Linguistica 4. It then creates an FSA to hold them all. The FSA is too big to look at in one piece, but we can look at parts of it.

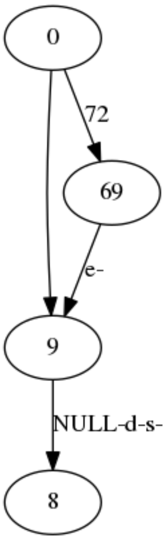


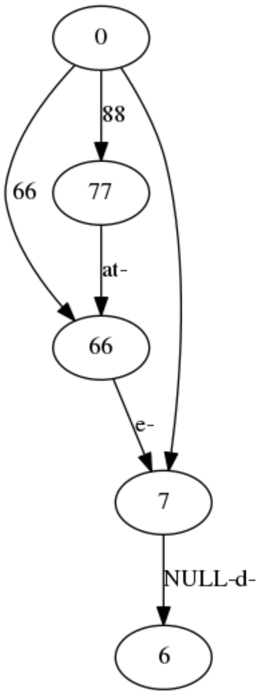





0
74
80
82
2
70
73
74
al-
ous-
ent-
5
NULL-ly-
4


0
126
70
154
84
92
56
67
66
82
152
49
31
ic-
38
81
45
4
36
ing-
ate-
44
ful-
80
nt-
ous-
al-
ive-
less-
4
-NULL-ly
1





Brown Corpus:

```
--------------------------------------------------------------
Words and their signatures
--------------------------------------------------------------
Word                          Signatures
--------------------------------------------------------------
```

'bout	['NULL-s']
'bouts	['NULL-s']
'long	['--NULL']
'long-	['--NULL']
'twould	["NULL-n't"]
'twouldn't	["NULL-n't"]
abash	['NULL-ed']
abashed	['NULL-ed']
absent	['NULL-ly']
absent-minded	['NULL-ness']
absent-mindedness	['NULL-ness']
absently	['NULL-ly']
absorbed	['ed-ing']
absorbing	['ed-ing']
abuse	['NULL-d']
abused	['NULL-d']
accompany-	['--ing']
accompanying	['--ing']
accomplish	['NULL-ed-ing']
accomplished	['NULL-ed-ing']
accomplishing	['NULL-ed-ing']
accord	['NULL-ing']
according	['NULL-ing']
account	['NULL-able']
accountable	['NULL-able']
accused	['ed-ing']
accusing	['ed-ing']
achieve	['--NULL']
achieve-	['--NULL']
acquire	['NULL-d-ment']
acquired	['NULL-d-ment']
acquirement	['NULL-d-ment']
actual	['NULL-ly']
actually	['NULL-ly']
admirable	['able-ation-ed-ers-ing']
admiration	['able-ation-ed-ers-ing']
admired	['d-rs', 'able-ation-ed-ers-ing']
admirers	['d-rs', 'able-ation-ed-ers-ing']
admiring	['able-ation-ed-ers-ing']
adorn	['NULL-ed']
adorned	['NULL-ed']
advance	['NULL-s']
advances	['NULL-s']
advantage	['NULL-s']
advantages	['NULL-s']

```
adventure                     ['NULL-s']
adventures                    ['NULL-s']
affair                        ['NULL-s']
affairs                       ['NULL-s']
affected                      ['ed-ion']
affection                     ['ed-ion']
again                         ['NULL-st']
against                       ['NULL-st']
aggravate                     ['NULL-d']
aggravated                    ['NULL-d']
agree                         ['NULL-able-d']
agreeable                     ['NULL-able-d']
agreed                        ['NULL-able-d']
aisle                         ['NULL-s']
aisles                        ['NULL-s']
alarm                         ['NULL-ed']
alarmed                       ['NULL-ed']
alley                         ['NULL-s']
alleys                        ['NULL-s']
allow                         ['NULL-ance-ed-ing']
allowance                     ['NULL-ance-ed-ing']
allowed                       ['NULL-ance-ed-ing']
allowing                      ['NULL-ance-ed-ing']


Analysis of each signature (for example:)

=============================================
NULL-ly

abrupt      absolute    according   accurate    adequate    annual
anxious     apparent    approximate awful       beautiful   bitter
blind       blunt       brief       brilliant   careful     casual
cautious    certain     chief       common      comparative conscious
consistent  continuous  curious     definite    deliberate  desperate
different   eager       earnest     economical  effective   efficient
emotional   enormous    entire      essential   eventual    evident
exact       exceptional exclusive   experimental extensive  financial
formal      former      fortunate   frequent    fundamental furious
generous    genuine     graceful    gradual     historical  hopeful
immediate   immense     impatient   important   increasing  independent
indirect    initial     instant     intense     literal     local
logical     loose       mental      mutual      natural     normal
obvious     occasional  original    painful     partial     particular
peculiar    permanent   physical    pleasant    political   positive
```

```
practical     precise       previous      principal     private       profound
prominent     prompt        proper        proportionate proud          quick
quiet         radical       rapid         recent        regular       repeated
reported      respective    rigid         rough         seeming       serious
severe        sharp         significant   silent        simultaneous  slight
smooth        solemn        special       spontaneous   stiff         strict
striking      subsequent    substantial   successful    sudden        sufficient
superb        supposed      surprising    swift         technical     thorough
thoughtful    tight         total         traditional   tremendous    typical
ultimate      unconscious   unexpected    unfortunate   unique        unlike
unusual       usual         utter         vague         vigorous      violent
vivid         wonderful
-------------------------
                                          Phono      Ordering       Total
Information in words if unanalyzed:       11610 +      16187 =      27797
  Information in words as analyzed:        6110 +        726 =       6836
Average count of top 5 stems: 357

-------------------------

High frequency possible affixes
Number of stems:  158
        al     weight:    74 count:    37
       ous     weight:    48 count:    16
         l     weight:    46 count:    46
       ent     weight:    45 count:    15
        nt     weight:    42 count:    21
       ate     weight:    33 count:    11
        us     weight:    32 count:    16
         t     weight:    31 count:    31
       cal     weight:    30 count:    10
         e     weight:    29 count:    29
        te     weight:    26 count:    13

=============================================
```

5.5 What is the question?

We identify morphemes due to frequency of occurrence: yes, but all of their sub-strings have at least as high a frequency, so frequency is only a small part of the matter; and due to the non-informativeness of their end with respect to what follows.

But those are *heuristics*: the real answer lies in formulating an FSA (with post-editing) that is simple, and generates the data.

5.5.1 Gibbs sampling

Word w is analyzed into morphemes $\{m-i\}$, indicated $\mathcal{M}$.
$M-ct(w)$: number of morphemes analyzed in word w (4 for *board ing house s*); this is the size of $\mathcal{M}$.
The length of morpheme m in symbols is indicated by $|m|$. The number of occurrences of morpheme m in the whole lexicon is $[m]$.

$$score = log(M - ct(w)) + \sum -m \in \mathcal{M} \frac{log(|m|!) + 5 \times |m|}{[m]} - log\, p(m)$$

morpheme	random	1 cycle	10 cycles	100 cycles
s	1639	1681	1253	1151
e	996	982	544	429
d	823	800	458	360
t	640	618	355	282
r	655	618	358	257
n	671	637	315	208
a	558	539	300	253
g	545	544	324	240
c	533	522	316	230
l	459	433	264	212
i	494	473	271	202
p	452	431	293	240
ing	235	461	1029	1059
’s	159	180	292	332
er	208	245	306	315
ed	431	532	640	631
-	45	–	102	363
es	241	289	277	262
re	174	211	242	287
ation	33	60	145	190
ness	26	134	154	154
able	27		140	174

random	1 cycle	10 cycles	100 cycles	200 cycles
board	board	board	board	board
board's	board's	board 's	board's	board 's
boarded	boarded	board ed	board ed	board ed
bo ar der	bo ar der	board er	board er	board er
boarding	boarding	boar ding	boar ding	board ing
boardi nghouses	boardi nghouses	boar ding houses	board ing houses	board ing house s
bo ards	bo ards	board s	board s	board s
boast	boast	boast	boast	boast
boasted	boasted	boasted	boast ed	boast ed
bo as tfully	bo as tfully	boastfully	boast fully	boast fully
boasting	boasting	boasti ng	boast ing	boa sting
bo a stings	bo a stings	boastings	boast ings	boast ings
boasts	boasts	boasts	boast s	boast s
boat	boat	boat	boat	boat
boat-y ard	boat-y ard	boat-yard	boat-year	boat-yard

5.5.2 Putting phonology into the lexicon

5.5.3 Putting segmentation structure in the lexicon: morphology 1

5.5.4 Successor Frequency

Zellig Harris 1955

5.6 What works better?

A better heuristic with about the same degree of simplicity is to look at word-final sequences of letters (if we are looking for suffixes), and evaluate them by multiplying their length times the number of times they occur. We will refer to this as the string's *robustness*. For a typical sample of written English of 14,000 words, we find the suffix ing occurring 961 times, and since its length is 3, that gives it a robustness score of 2,883. The second most robust word-final sequence in this corpus is s, which occurs 2,778 times, and thus has a robustness score of 2,778.

Figure 5.5.1 Successor frequency

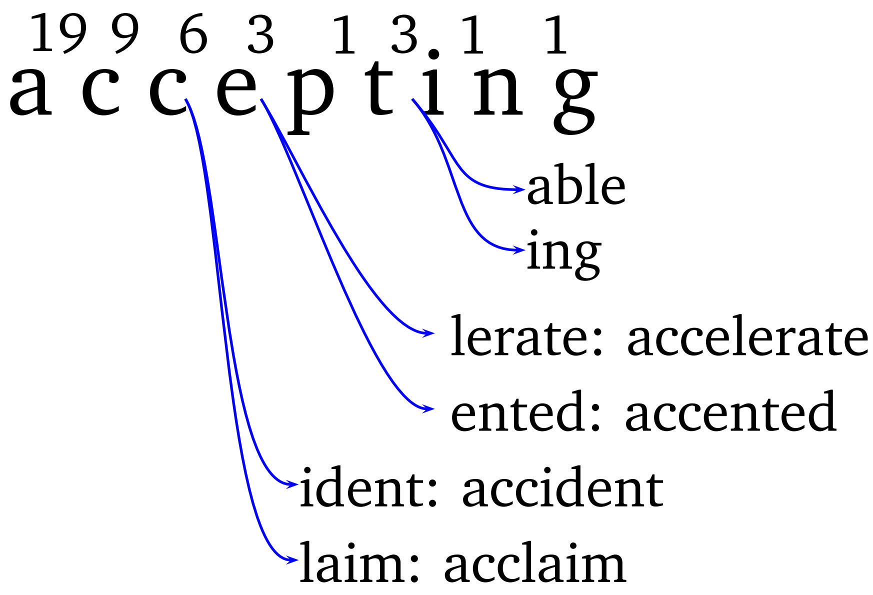
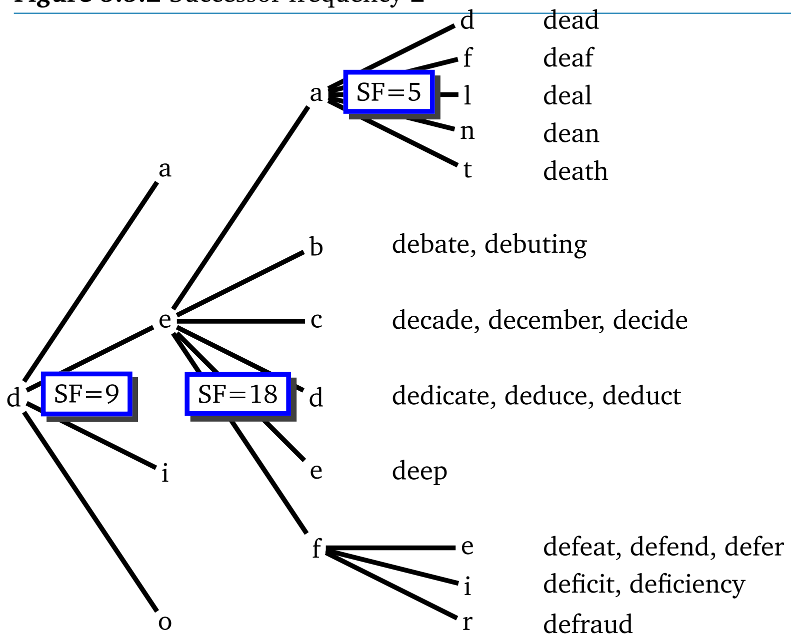


Figure 5.5.2 Successor frequency 2



5.7 adding layers of morphology

An initial morphology of the suffixes of English produces a very simple FSA. [example]

We ask each edge that is associated with a large set of stems to advance a set of candidates of stem-final suffixes, based on the count and the length of these candidate strings. For the stems that appear before *NULL-ly*, we obtain the following FSA:

Let us look at the morphemes associated with some of the edges. Edge 126, in the top left corner, contains the following labels (stems). The ones in blue are surely correct; the shorter ones, like *eth-* or *com-* are probably incorrect.

Edge number 126 To state: 67					
method	mag	log	ecolog	ideolog	psycholog
chronolog	graph	geograph	philosoph	eth	com
anatom	mechan	clin	cyn	typ	numer
categor	rhetor	histor	class	mathemat	tact
theoret	polit	uncrit	skept	vert	statist
analyt	paradox				

These are all analyzed as appearing before the suffix -*c*, and then -*al*, and then either followed by nothing or by *ly*.

Edge 66 is associated are stems that do not end in -*c*, but are followed by -*al*, and then either followed by nothing or by *ly*:

Edge number 66 To state: 36 Stem					
unequivoc	fisc	judici	unoffici	artifici	superfici
substanti	exponenti	quintessenti	potenti	sequenti	dism
phenomen	nomin	occasion	provision	congression	education
gravitation	fraction	addition	condition	uncondition	intention
convention	exception	proportion	unconstitution	etern	intern
cerebr	bilater	liter	sever	architectur	structur
accident	incident	coincident	increment	horizont	continu
usu	factu	contractu	perpetu	habitu	conceptu

How does this get produced? Here is an ordered list of the first 10 morphemes that are pulled out by this strategy:

Order:	From state:	Edge number	To state:	morpheme
1	20	37	2	er
2	21	39	2	tion
3	22	41	2	ing
4	23	43	5	e
5	24	44	6	e
6	25	46	2	ment
7	26	48	7	s
8	27	49	2	ist
9	28	51	24	at
10	29	53	2	ian

Let's look at the first morphemes that are specifically pulled out of the stems that precede NULL.s:

Order:	From state:	Edge number	To state:	morpheme
1	20	37	2	er
2	21	39	2	tion
3	22	41	2	ing
6	25	46	2	ment
8	27	49	2	ist
10	29	53	2	ian
11	30	55	2	tor
13	32	59	2	on
16	35	65	2	le
22	41	77	2	nce
23	42	79	2	nt
24	43	81	2	te
27	46	87	2	re
29	48	91	2	al
36	55	103	2	ne
37	56	105	2	et
39	58	109	2	ic
41	60	113	2	ship
42	61	115	2	out
44	63	119	2	de
45	64	121	2	ard
47	66	125	2	tive

The first set of stems has pulled off *-er* as a suffix on 540 words. In the following table, stems in blue are correct, and stems in green are arguably correct, though the vast majority of them are of the form *noun-verb-er*, where the noun is the object of the verb (as in *bartender*). Some cases are less regular: a *biographer* is not someone who biographs, but rather someone who writes biographies; but analyzing *biograph-er* seems perfectly reasonable.

Edge number 66 To state: 36 Stem					
scrubb	limb	climb	bomb	cucumb	plumb
trac	ulc	danc	announc	enforc	sauc
ringlead	cheerlead	load	grad	crusad	invad
shredd	feed	breed	raid	spid	provid
weld	homebuild	shipbuild	guild	fold	cardhold
stakehold	debthold	unithold	mold	bould	land
highland	island	salamand	command	bystand	defend
gend	spend	contend	bartend	bind	cind
remind	grind	transpond	decod	schrod	forward
camcord	intrud	auctione	conventione	overse	waf
coff	counteroff	lif	aquif	golf	surf
villag	teenag	pag	arbitrag	voyag	bridg
rodg	dagg	digg	jogg	mugg	folg
rang	strang	messeng	harbing	gunsling	ring
wing	charg	cheeseburg	hamburg	lug	bleach
schoolteach	ranch	launch	crunch	dispatch	watch
vouch	biograph	demograph	photograph	goph	philosoph
wash	dishwash	finish	extinguish	push	math
fanci	pacifi	amplifi	clothi	ski	chandeli
fli	highfli	colli	copi	photocopi	barri
couri	hoosi	dossi	fronti	courti	sneak
break	shak	lak	peacemak	pacemak	troublemak
dealmak	filmmak	carmak	moneymak	tak	caretak
hack	pack	meatpack	crack	firecrack	track
woodpeck	traffick	kick	slick	stick	knickerbock
block	rock	suck	seek	bik	hik
strik	talk	tank	think	drink	bunk
onlook	mark	casework	cowork	york	hawk
heal	gambl	assembl	recycl	peddl	toddl
swindl	feel	jewel	muffl	juggl	smuggl
mail	trail	fil	oil	sprinkl	install
resell	booksell	bestsell	tell	dwell	zell
kill	painkill	drill	thrill	roll	stroll
school	stapl	sampl	wrestl	hustl	settl
haul	rul	trawl	bowl	guzzl	dream
fram	ibm	disclaim	tim	programm	glimm
swimm	somm	drumm	newcom	monom	astronom
inform	perform	transform	polym	clean	afrikan
open	sweeten	fasten	listen	campaign	sign
bargain	complain	train	retain	entertain	din
berlin	airlin	jetlin	marin	bann	scann
beginn	spinn	sinn	forerunn	parishion	pension
practition	petition	question	common	soon	earn
northern	southern	eastern	western	midwestern	burn
vintn	kindergartn	down	landown	skyscrap	beep
peacekeep	housekeep	gatekeep	bookkeep	innkeep	shopkeep

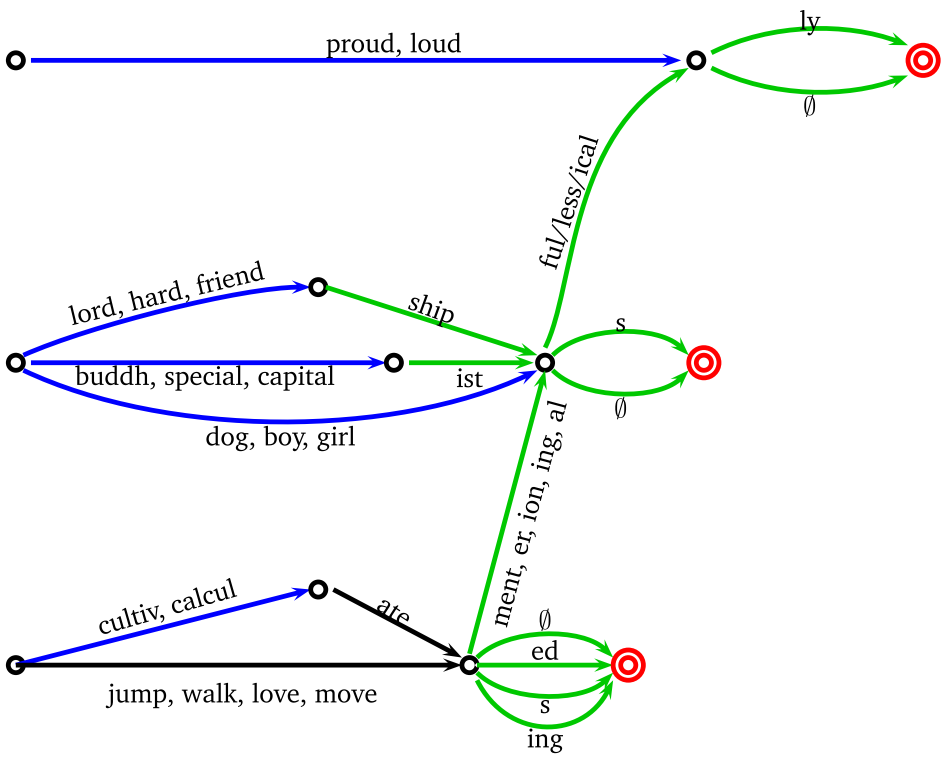
Edge number 66 To state: 36 (continued)					
minesweep	snip	junip	wip	help	camp
jump	interlop	troop	paratroop	rop	handicapp
rapp	wrapp	shipp	clipp	flipp	stripp
whopp	stopp	casp	jasp	bear	wear
murder	suffer	gather	cater	adulter	admir
labor	scor	explor	reinsur	lectur	adventur
las	rais	fundrais	apprais	exercis	merchandis
cruis	cleans	dispens	endors	pass	hairdress
accus	trous	heat	sweat	skat	float
floodwat	backwat	street	cathet	diet	telemarket
paramet	millimet	centimet	odomet	kilomet	thermomet
interpret	raft	draft	freight	fight	firefight
granddaught	stepdaught	wait	arbit	typewrit	songwrit
screenwrit	sportswrit	scriptwrit	copywrit	recruit	smelt
supercent	rent	dissent	point	headhunt	discount
scoot	shoot	adapt	chapt	helicopt	start
comfort	support	transport	frankfurt	forecast	postmast
roast	toast	disast	mobst	semest	forest
harvest	gangst	youngst	canist	pollst	hamst
rost	dumpst	bust	dust	adjust	platt
gett	sett	hitt	transmitt	critt	sitt
spott	cutt	gutt	putt	stutt	pollut
telecommut	minicomput	microcomput	supercomput	rescu	leagu
sav	lifesav	believ	reliev	nev	waiv
sliv	cabdriv	solv	revolv	holdov	changeov
hangov	rollov	mov	turnov	leftov	layov
observ	draw	review	interview	skew	widow
whistleblow	wildflow	sunflow	follow	mow	superpow
mix	box	ballplay	pay	ratepay	pray
moy	destroy	dry	fry	blaz	freez
stabiliz	fertiliz	tranquiliz	organiz	appetiz	bulldoz
analyz					

The second set of stems is this, based on a suffix *-tion*:

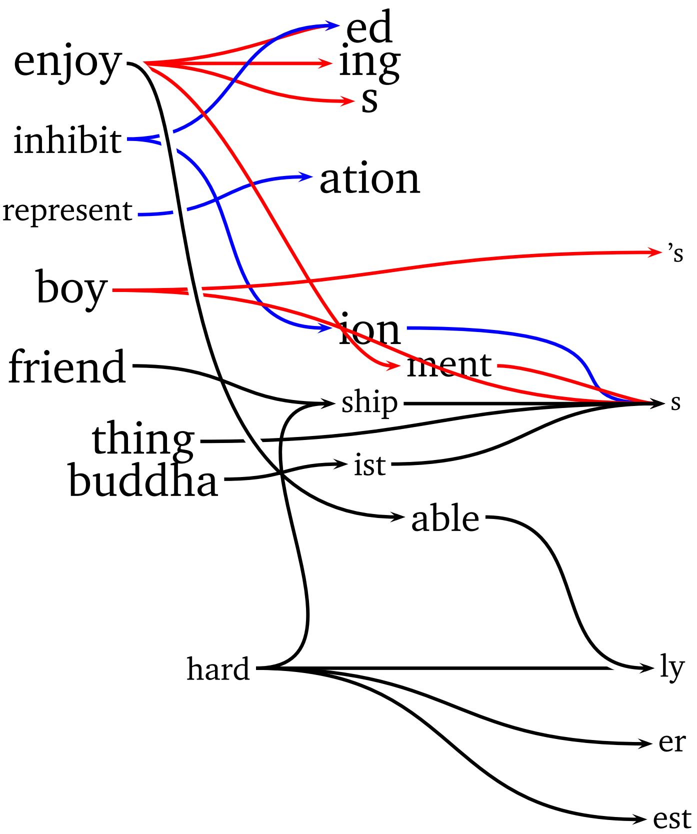
Edge number 66 To state: 36 Stem					
perturba	medica	indica	syndica	specifica	modifica
amplifica	magnifica	clarifica	classifica	identifica	certifica
implica	complica	applica	fabrica	loca	reloca
disloca	provoca	depreda	consolida	liquida	recommenda
delega	allega	obliga	interroga	denuncia	affilia
varia	appropria	negotia	renegotia	devia	abbrevia
revela	installa	cancella	viola	transla	specula
miscalcula	circula	regula	simula	formula	manipula
popula	congratula	proclama	exclama	affirma	confirma
transforma	explana	designa	resigna	combina	vaccina
origina	machina	inclina	examina	elimina	recrimina
denomina	termina	determina	rumina	assassina	destina
incarna	participa	preoccupa	declara	prepara	separa
vibra	delibera	reverbera	considera	exaggera	altera
aspira	expira	collabora	decora	perfora	explora
aberra	arbitra	concentra	registra	demonstra	illustra
configura	accusa	expecta	interpreta	cita	solicita
imita	limita	consulta	planta	presenta	misrepresenta
connota	quota	adapta	tempta	flirta	exhorta
manifesta	infesta	worksta	muta	reputa	amputa
valua	evalua	devalua	insinua	equa	fluctua
depriva	ova	renova	innova	observa	reserva
nationaliza	rationaliza	liberaliza	generaliza	capitaliza	hospitaliza
reorganiza	immuniza	characteriza	authoriza	dramatiza	privatiza
infrac	contrac	abstrac	distrac	attrac	defec
imperfec	rejec	injec	projec	selec	reflec
recollec	connec	interconnec	inspec	intersec	contradic
predic	afflic	depic	restric	evic	convic
injunc	concoc	abduc	deduc	reduc	reproduc
dele	comple	secre	inhibi	prohibi	exhibi
edi	rendi	precondi	defini	admoni	deposi
disposi	exposi	repeti	supersti	tui	deten
absten	atten	inven	lo	no	po
decep	misconcep	percep	mispercep	intercep	subscrip
prescrip	inscrip	redemp	exemp	assump	adop
interrup	disrup	asser	exer	por	distor
sugges	contribu	distribu	solu	resolu	substitu

Edge number 22 To state: 13 Stem
describ prescrib surfac outpac embrac balanc distanc experienc silenc sentenc influenc denounc persuad pervad cor

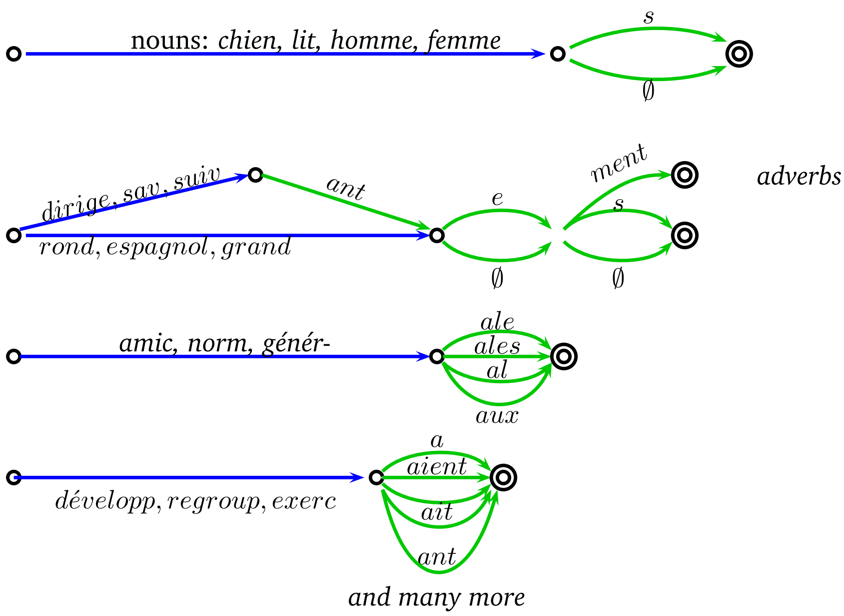
5.8 Immediate issues: getting the morphology right



English morphology: morphemes associated with nodes of an FSA



French



5

Signatures	Exemplar	Descr. Length (model)	Corpus Count	Stem Count	Source
NULL-s	accommodation	12996.7	13787	978	SF1
's-NULL	a*a*u	4237.23	8263	324	SF1
NULL-ly	according	3436.6	3391	259	SF1
NULL-ed-ing-s	account	886.936	2852	76	SF1
ed.ing	allott	1036.02	272	71	SF1
NULL.ed	abolish	1308.03	392	91	SF1
NULL.ed.s	accent	646.789	859	51	SF1
NULL.ing.s	boat	592.372	1060	46	SF1
NULL.ing	abound	1078.03	528	76	SF1
NULL.ed.ing	absorb	503.885	364	37	SF1
ing.s	awaken	172.814	29	11	SF1
ed.ing.s	fad	56.9268	13	3	SF1
's-NULL-s	afternoon	967.65	4258	83	SF1
e-ed-es-ing	accus	480.75	1345	40	Known stems to
e.ed.es	advanc	497.055	702	38	Check sigs
e.ed	acquiesc	825.969	311	58	Check sigs
e.ed.ing	anticipat	337.05	189	24	Known stems to
e.es.ing	battl	208.905	478	16	Known stems to
e.ing	abid	395.385	128	27	SF1
ed.es	aggravat	330.992	146	23	Check sigs
es.ing	celebrat	254.894	72	17	SF1
ed.es.ing	experienc	55.0602	35	3	From known sten
ies-y	abilit	899.932	642	66	SF1
NULL-al-s	addition	310.116	485	24	SF1
NULL.al	dramatic	87.2327	65	6	Check sigs
NULL-ly-s	absolute	320.709	468	25	SF1

1. Real versus accidental subcases: When should sub-signatures be subsumed by the "mother" signature? When are two signatures two samples from the same multinomial distribution? In some cases, this seems like a question with a clear meaning, as in case (a). Case (b) is less clear. Case (e) is interestingly different.

2. NULL-s *vs* NULL.ed.ing.s;

3. NULL-s *vs* NULL-s-'s

4. NULL-ed-ing-s *vs* NULL-ed-ing-ment-s

5. NULL-ed-er-ers-ing-s: how do we treat this?

6. NULL-ed-ing-s (vs) NULL-ing-s (e.g., *pull-pulling-pulls*); similar question arises for all so-called *strong* English verbs (this is a linguistically common situation).

7. The role of "post-editing": phonology and morphophonology. [6]

8. final *e*-deletion in English

9. C-doubling (*cut/cutting, hit/hitting; bite/bitten*)

10. *i/y* alternation: *beauty-beatiful; fly/flies;*

[5]**English**: NULL - s - ed - ing - es- er - 's - e - ly - y - al - ers - in - ic - tion - ation - en - ies - ion - able - ity - ness - ous - ate - ent - ment - t (*burnt*) - ism - man - est - ant - ence - ated - ical - ance - tive - ating - less - d (*agreed*) - ted - men - a (*Americana, formul-a/-ate*) - n (*blow/blown*) - ful - or - ive - on - ian - age - ial - o (*command-o, concert-o*) ...

[6]**French**: s - es - e- er - ent - ant - a - ée - é - és - ie - re - ement - tion - ique - ait - èrent - on - ées - te - ation - is - aient - al - ité - eur - aire - it - isme - en - age - ion - aux - ier - ale - iste - ien - t - eux - ance - ence - elle - iens - euse - ants - ienne - sion ...

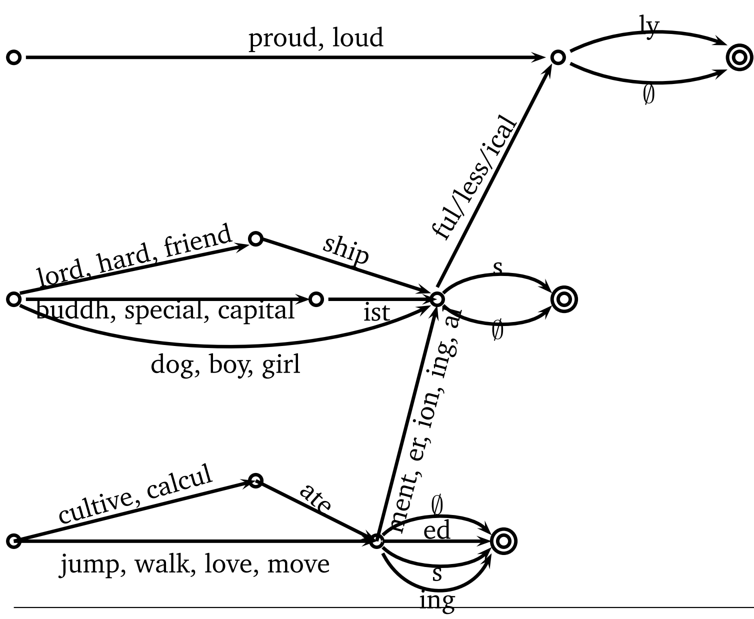
A calculation regarding a conjectured "phonological process" that falls half-way between heuristic and application of our DL-based objective function: Consider a process described as mapping $X \rightarrow Y/context$. [7] Rewrite the data as if that expressed an equivalence: we "divide" the data by that relation (for simplicity's sake, we ignore the context). [8] In this case, the result is a corpus from which all e's have been deleted. [9] What is the impact on the morphology that is induced from this new data? The lexical items are (of course) simpler (shorter). But the new morphology is *much* simpler than before, because *signatures* now collapse. *NULL.ed.ing.s* and *e.ed.es.ing* both map to *NULL.d.ing.s*. Each was of roughly the same order of magnitude; hence the bit cost of a pointer to the new signature is 1 bit less than that of the previous pointers, and that is a single bit of savings multiplied by thousands of times in the description length of the new corpus (quite independent of the missing es).

11. Succession of affixes: Stems of the signature NULL-s end in *ship, ist, ment, ing*. We can apply the analysis iteratively, re-analyzing all stems (and unanalyzed words), but this is not an adequate solution.

12. *NULL-ed-ing-s* vs. *t-ted-ts-ting* (Faulty MDL assumption?)

13. Clustering when no stem samples all its possible suffixes, but a family of them does: verbs in Romance languages.

Figure 5.8.1 What we would like to generate



[7] $e \rightarrow \emptyset / -ed, -ing$
[8] $corpus \Rightarrow corpus/e \approx \emptyset$.
[9] *creeps* is now spelled *crps*, and *creeping* is *crping*.

Figure 5.8.2 Top signatures: First set

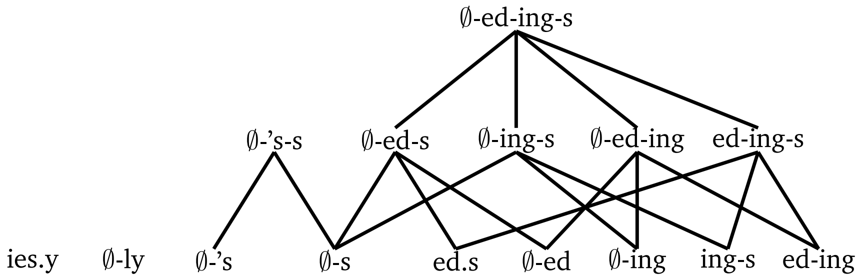
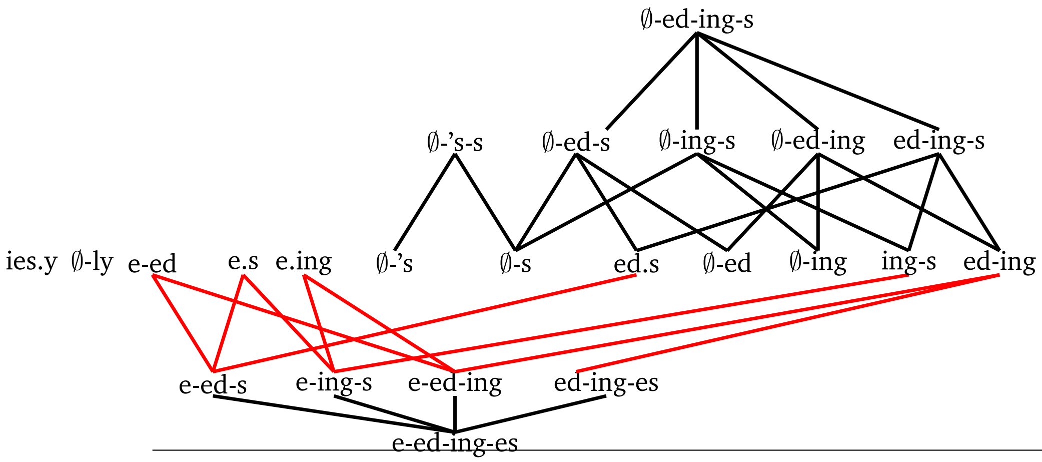


Figure 5.8.3 3 Top signatures: inverted



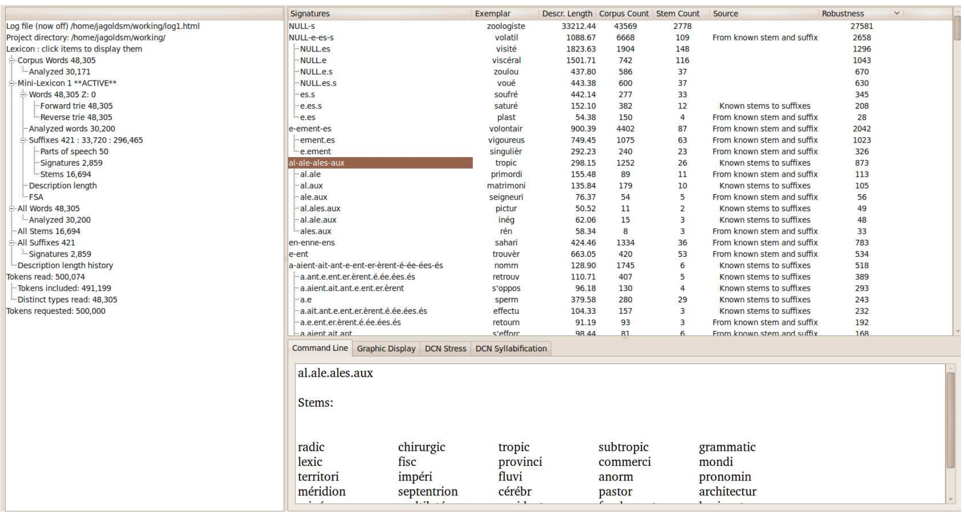
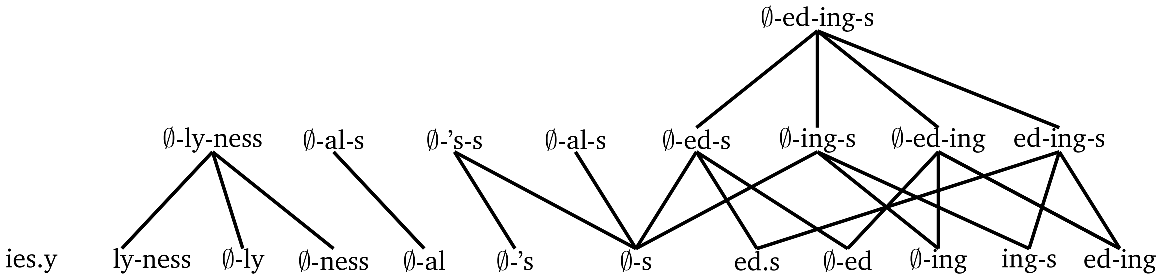
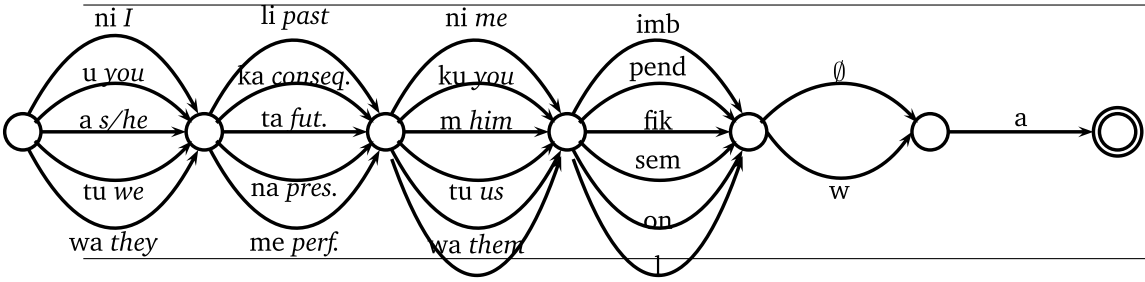


Figure 5.8.4 Stage 4



5.9 Swahili

Figure 5.9.1 Simplified Swahili verbal morphology



Typical case where morpheme frequency is more important than a count of the number of letters, in determining description length. The following is a correct change that this DL computation gets right:

$$ak + \{a, i\} + \{stems\} \rightarrow a + \{ka, ki\} + \{stems\}$$

because *ak* occurs nowhere else, but *ka* and *ki* are common. What is important is global, rather than local, parsimony.

5.9.1 String Edit Distance

5.9.2 Rich morphologies : morphology 2

Grammatical distribution 6

6.1 Week 8: From neighbors to categories

This chapter, which describes work I have done with Wang Xiuli, describes some explorations of how words of a natural language are located in a high-dimensional space when the distance between individual pairs of words is based, directly or indirectly, on the number of syntactic contexts the two words share. From the point of view of the algorithms which we use, the work is based on methods explored by Niyogi, Belkin, and quite a few others, methods that use graph-theoretic notions in order to define and determine a manifold of relatively low-dimensionality that lies reasonably closely to most of a large set of observed data points. From the point of view of the linguistic question involved, the work is intended to develop a data-driven method that can be used on virtually any language in order to create a geometrical object which can be visualized by a human, and which can be used to give a rough account of the syntactic— or, more specifically, distributional—properties of words.

6.1.1 Thought flow

The train of thought here involves a number of steps, and several independent decisions.

1. We begin with a corpus, and a decision to use information that we can get from it, which is often called "distributional information".

2. One way is to define properties by contexts. A context is a specification of the words occurring in a particular relation to the word we care about. For example, we could define the context "the —" as the context "occurring immediately after the word *the*." Then any word which appears there *possesses* that property.

3. We can define relational properties, which are possessed by pairs of words (word-types). For example, we can define the common contexts of two words as the intersection of their individual contexts.

4. We can measure the linkedness of two words by the size of the common contexts of the two words. This is symmetrical, of course, and it is heavily influenced by the frequency of each of the individual words.

5. We can immediately visualize this linkedness as an undirected weighted graph, in which each node corresponds to a word, where each edge connecting two nodes (words) corresponds to a non-zero count of the number of contexts shared by the two words. Let's assume that we have a convenient way to number our words, so we can talk about "$word_1, word_2, \ldots word_{50,000}$," or "$w_1, \ldots$" for short. Let's suppose that there are 50,000 distinct words in our corpus.

6. Whenever we think about an undirected graph, we also think of a symmetric matrix with zeros down the major diagonal, with one row and column for each node, and a value $m_{i,j}$ equal to the edge weight we just discussed. This is called the graph's adjacency matrix. The rows and columns of the matrix each correspond to a word, and we'll use the same numbering as above (for word w_1, etc.).

7. The eigenvectors of a symmetric matrix **M** are all real, and when they're all positive, it's natural to think of the matrix as defining an ellipsoid. There are two different, but not very different, ways of visualizing this. You could imagine a sphere S in n-space, the set of points exactly distance 1 from the origin, and then visualize the image of that sphere under the effect of the matrix: the set of all points **Mv** where $|\mathbf{v}| = \mathbf{1}$. The other way is more common, actually, and that is to visualize the set of vectors for which the so-called Rayleigh quotient is 1. The Rayleigh quotient is the inner product of a vector and its image under the matrix (divided by the norm of the vector, if you are not willing to restrict yourself to vectors of unit norm): $R(M, v) = \frac{(v, Mv)}{|v|}$. It is often discussed in the case of vector spaces over the complex numbers, and in that case we think about hermitian rather than symmetric matrices: $m_{i,j}$ must be the complex conjugate of $m_{j,i}$. These matrices have real eigenvalues.

8. The various axes of these ellipsoids point in the directions of the eigenvectors of the matrix M.

9. Rather than look at M, however, we typically look at the closely related matrix L (for Laplacian). We define first the diagonal matrix D, for which the (i,i)th element $d_{i,i} = \sum_j m_{i,j}$. Then the Laplacian is defined as D-M. Hence it is identical to D down the major diagonal, and its rows and columns all sum to 0 (and it is symmetrical).

10. Since we care about properties of words that are largely indendent of frequency, we are more interested in one of the normalized forms of the Laplacian. Chung has emphasized the relevance of the normalized Laplacian $\mathcal{L}$, which is obtained by pre- and post-multiplying L by $D^{-\frac{1}{2}}$. The major diagonal of the normalized Laplacian is all '1', but the columns and rows do not sum to zero.

11. It is quite amazing that when we minimize the Rayleigh quotient, we also minimize an expression that we can interpret as a test for a good embedding of words in R^n that respects the linkedness of the graph. Suppose we compute the first 10 eigenvectors of normalized Laplacian (those with the lowest positive eigenvalues). Each of those eigenvectors assigns a real number to each word; that real number is the coordinate of the eigenvector of the coordinate corresponding to the word in question. (Got that?)

12. Consider the eigenvector with the smallest positive eigenvalue. Its coordinates consist of a real number that can be associated with each of the words w_i. They can be thought of as instructions for placing each word along a real number line. This eigenvector has the property that it assigns the lowest possible "discrepancy" between the placement of words on a real line and the linkedness of the same words in the original graph that started this whole process going. The discrepancy is defined as the sum (over all of the words) of the product of $(v_i - v_j)^2 \times m_{i,j}$.

13. That lowest eigenvector spans a 1-dimensional space in our original space of 50,000 dimensions. We look now at the orthogonal complement, which leaves us in a space of dimensionality 49,999. The next eigenvector (with the next smallest positive eigenvalue) will be the one that assigns coordinates to the words in a way that minimizes the discrepancy (same discrepancy as above), in a direction that is (as we have said) orthogonal to the previous eigenvector. That gives us a second coordinate for each of the 50,000 words.

14. We can continue doing this until we decide we have enough coordinates —10, let's say. This gives us an embedding of our vocabulary in R^{10}.

15. Unfortunately, there is no inherent meaning to distance or direction in this space. That is, given word 1, we can say whether word 2 or word 3 is closer to it, and we can rank the k closest words to a given word, but measurable closeness in one part of the space does not naturally transform to closeness in some other part of the space.

16. For this reason, we only use this embedding for one purpose: to allow us to speak of the k-nearest neighbors to any particular word. And then we construct graphs of this sort, and look at them with Gephi, and various clustering techniques can be applied to it as well. We discuss these in sections 6.2 and **??**.

6.1.2 Initial similarity measure

Much recent work has been motivated by the relative ease with which a large amount of data can be comfortably handled computationally, even when the scientist has the prior intuition that only a small subpart of the data is likely to play an important role in answering the questions he is interested in. If we take the notion of syntactic part of speech of a word w to be a rough approximation to a set of categories describing the syntactic distributional properties of w, then some subset of features such as the following should be useful.

Property			
W(-1)	=	w_j	means the word to the immediately left of w is w_j;
W(1)	=	w_j	means the word to the immediately right of w is w_j;
W(-2)	=	w_j	means the word that is two words the of w is w_j; etc.
W(-2,-1)	=	(w_j,w_k)	means W(-2)=w_j and W(-1)=w_k.
W(-1,1)	=	(w_j,w_k)	means W(-1)=w_j and W(1)=w_k.

With all of our experiments described below, we have used the three features W(-2,-1), W(-1,1), and W(1,2). Thus, in a corpus consisting exactly of the first sentence of this paper, the word *explorations* would be assigned three features: (describes, some); (some,of); and (of,how).

Let V be the number of distinct word types in the language. Then there are in principle V^2 features of the type W(-2,-1), and also of the type W(-1,1) and W(1,2). But the number of such features that are actually used is a small subset of the total number.

We define $f(w_i, w_j)$ as the number of distinct features (using the contextual features just defined) shared by words w_i and w_j. It's natural to think of a graph now in which the nodes are our words, and the edges are weighted by $f(w_i, w_j)$. The $laplacian$ of that graph is defined as the matrix$M$ in which $M(i, j) = f(w_i, w_j)$ when $i \neq j$; in the case of the diagonal elements, we define $d(i)$ as $\sum_{k \neq i} M(i, k)$, and then $M(i, i)$ is defined as $-1 \times d(i)$. (In this case, $d(i)$ measures the frequency of the i^{th} word.)

(We now have an initial similarity measure between words, but this similarity is not normalized for frequency: high frequency words will be much more similarity to others words that low frequency words will. Even if we normalize for frequency, though, the simplest ways of estimating similarity of distribution between two words on the basis of this data—using the cosine of the angle subtended by vectors pointing to each of the two words—is not as good as we might hope.)

6.1.3 Normalized laplacian

A number of researchers have explored the idea of taking a large set of data in a space of very high dimensionality, and finding a subspace of much lower dimensionality which is almost everywhere fairly close to the data. We've been especially influenced by the work of Partha Niyogi and Mikhael Belkin in the discussion that follows.

In this case, this means finding the eigenvectors of a normalized version of the graph laplacian. The normalized version of M, which we call N, is defined as follows: for all $i, N(i, i) = 1$, while for $(i, j), i \neq j$, we use the $d()$ function defined above to normalize, and say that $N(i, j) = \frac{M(i,j)}{\sqrt{d(i)d(j)}}$.

We computed the first 11 eigenvectors of this normalized laplacian—those with the lowest eigenvectors, and used the 2nd through the 11th to give us coordinates for each word. Each word is

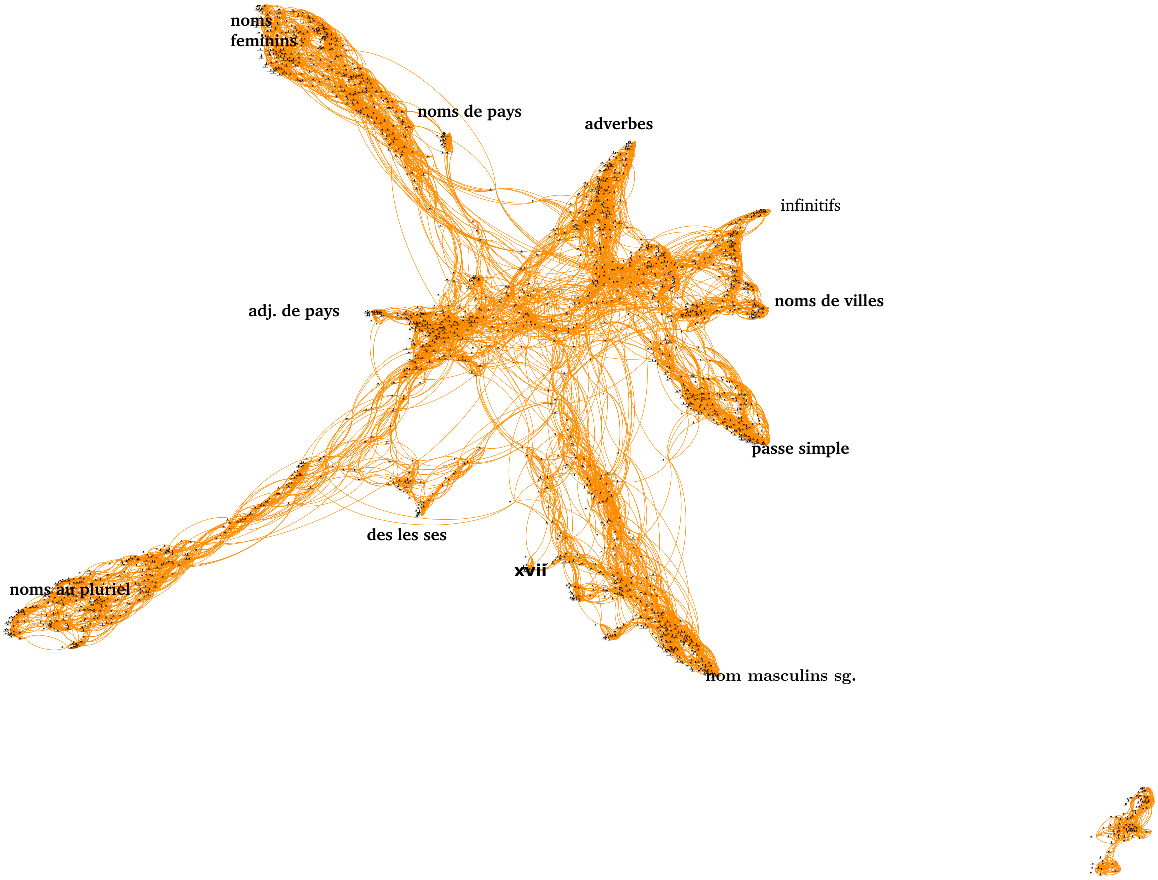


Fig. 6.1: 2,000 words French

thus associated with a point in R^{10}. We then select, for each word, the k closest words to it in this new space. These are the neighbors that we will explore below.

6.2 Visualization