Structure Learning in Weighted Languages

András Kornai, Attila Zséder, Gábor Recski
HAS Computer and Automation Research Institute
H-1111 Kende u 13-17, Budapest
{kornai,zseder,recski}@sztaki.mta.hu

Abstract

We present Minimum Description Length techniques for learning the structure of weighted languages. MDL is already widely used both for segmentation and classification tasks, and here we show it can be used to formalize further important tools in the descriptive linguists' toolbox, including the distinction between accidental and systematic gaps in the data, the detection of ambiguity, the selective discarding of data, and the merging of categories.

Introduction

The Minimum Description Length (MDL, see Rissanen 1978) framework is primarily about data compression: if we are given some data D, our goal is to find a model $\mathcal{M}$, and a correction term $\mathcal{E}$, such that the model output and the correction term together describe the data, and transmitting $\mathcal{M}$ and $\mathcal{E}$ takes fewer bits than transmitting any competing $\mathcal{M}'$ and $\mathcal{E}'$.

From the very beginning, starting with Pāṇini, linguists have put a premium on brevity. The hope is that the shortest theory is the best theory (see Vitanyi and Li 2000), at least if we are willing to posit a theory of Universal Grammar (UG) that will let us specify $\mathcal{M}$ briefly, since we can assume UG to be amortized over many languages.

In this paper we study the problem of compressing weighted languages by presenting them via weighted finite state automata (WFSA). The theoretical approach we discuss here has a long history: the founding paper of Kolmogorov complexity, Solomonoff (1964), already studied the problem of inferring a grammar from data, and Grünwald (1996) uses MDL to infer CFGs from corpora, there conceived of as long strings over a finite alphabet. It is fair to say that this theory has not had much impact on computational practice, where grammatical inference is dominated by the standard n-gram based language modeling methods, see Jelinek (1997) for an excellent summary of the basic ideas and techniques, most of which are still in wide use.

While the two approaches may coincide in certain cases (see Grünwald 1996), and in theory n-gram models are just a special case of the general WFSA, in practice they are divided by a fundamental difference in modeling unseen data. From an engineering standpoint, Church et al (2007) are entirely right in saying:

> No matter how much data we have, we never have enough. Nothing has zero probability.

Linguists, starting perhaps with Chomsky (1965), draw a bright line between *accidentally* and *systematically* missing data, and would prefer to restrict backoff techniques to the accidental gaps. The distinction is often lost in applied work, because the models need to be built in a noisy environment, where frequent typos like **teh* and similar performance errors can easily overwhelm genuine items like *boisterous* or *mopeds* by an order of magnitude or even more. In the eyes of many linguists, this observation alone is sufficient to rob probabilistic models of grammatical content, since this makes it impossible to define a single threshold g such that all and only strings with weight greater than g are grammatical.

Aside from this subtle but important distinction between accidental and systematic gaps, both kinds of language modeling can be cast in the same formal terms: we fit a model $\mathcal{M}$ that minimizes some function E (typically, the squared sum) of the error $\mathcal{E}$. Obviously, the more parameters $\mathcal{M}$ has, the better fit we can obtain. Much of contemporary computational linguistics follows the route of training simple models such

as Hidden Markov Models (HMMs) and probabilistic context-free grammars (PCFGs) with very many parameters, and stops adding more only when compressing the memory footprint is of paramount importance. As (Church et al., 2007) notes, applications like the contextual speller of Microsoft Office simply could not ship without keeping the language model within reasonable size limits. In such cases, we are quite willing to trade in E for gains in the size M of $\mathcal{M}$, and considerations of optimizing the sum of the two are simply irrelevant.

In contrast, our strategy is to search for model which measures both $\mathcal{M}$ and $\mathcal{E}$ in bits, and optimizes the sum $M + E$, not because we put such a premium on data compression, but rather because we follow in Pāṇini's footsteps. Our goal is finding *structural* models capable of distinguishing structurally excluded (ungrammatical) strings like *furiously sleep ideas green colorless* from low probability but grammatical strings like *colorless green ideas sleep furiously* (Pereira, 2000). For this more ambitious goal comparing models with different number of parameters is a key issue, and this is precisely where MDL is helpful.

The rest of this Introduction provides the basic definitions, notation, and terminology, all fairly standard except for the use of Moore rather than Mealy machines – the significance of this choice will be discussed in Section 2. In Section 1 we bring a fundamental idea of signal processing, *quantization error*, to bear on the problem of model selection, illustrating the issue on a real example, the proquant system of Hungarian. In Section 2 we show how one of the most powerful tools at disposal of the linguist, *ambiguity*, can be detected by MDL, bringing another standard idea, *signal to noise ratio* to bear. In Section 3 we discuss another real example, Hungarian morphotactics, and show that two methods widely (but shamefacedly) used in practice, discarding data and merging descriptive categories, can be used on a principled basis within MDL. Our goal is to show that by consistent application of MDL principles we can automatically set up the kind of models that linguists would set up. Ultimately, both man and machine work toward the same goal, optimization of grammar elegance or, what is the same, brevity.

Definition 1. Given some finite alphabet Σ, a *weighted language* p over this alphabet is defined as a mapping $p : \Sigma^* \to \mathbb{R}$ taking non-negative values such that $\sum_{\alpha \in \Sigma^*} p(\alpha) = 1$. This is less general than the standard notion of noncommutative power series with weights taken in arbitrary semirings (Eilenberg 1974, Salomaa 1978) but will suffice here. The stringset $\{\alpha | p(\alpha) > 0\}$ is called the *support* of p and will be denoted by $S(p)$.

Definition 2. Given two weighted languages p and q, we say the Kullback-Leibler (KL) *approximation error Q* of q relative to p is $\sum_{\alpha \in S(q)} p(\alpha) \log(p(\alpha)/q(\alpha))$. The *entropy* of p is defined as $-\sum_{\alpha \in S(p)} p(\alpha) \log(p(\alpha))$.

Definition 3. A WFSA $\mathcal{M}$ is defined by a square transition matrix M whose element m_{ij} give the probability of transition from state i to state j, an emission list h that gives a string $h_i \in \Sigma^*$ for each $i \neq 0$, and an acceptance vector $\vec{a}$ whose i-th component is 1 if i is an accepting state and 0 otherwise. There is a unique initial state which starts the state numbering at 0, and we permit states with empty outputs. Rows of M must sum to 1. Thus we have defined WFSA as normalized probability-weighted nondeterministic Moore machines.

Definition 4. The *weight* a WFSA assigns to a generation path is the product of the weights on the edges traversed, and the weight it assigns to a string α is the sum of the weights assigned to all paths that generate α.

1 Quantization error

The notions of quantization error and quantization noise, while well known in the signal processing literature (for a monographic treatment, see Widrow and Kollár 2008), and widely used in speech processing (Makhoul et al., 1985), have had little impact on language processing. Yet MDL description of even the simplest weighted language brings up a significant problem that cannot be addressed without approximation.

Let p be a non-computable real number between 0 and 1, and let us define the language A as containing only two strings, a and b, with probability p and $1 - p$ respectively. Since p is incompressible, the only way Alice can send A to Bob is by sending all bits of p. By Alice sending only the first n bits, Bob obtains a language A_n that approximates A with error of 2^{-n}. Since sending the strings $\{a, b\}$ has only a small constant cost, the overall MDL cost is dominated by the error term E, which is just as incompressible as the original p was.

As long as the weights themselves are treated as information objects of arbitrary capacity, there is no way out of this conundrum (de Leeuw 1956). On the other hand, the weighted languages we encounter in practice are generally abstracted from gigaword or smaller corpora, and as such their inherent precision is less than 32 bits. For weighted languages with finite support (corpora and language models without smoothing) p is simply a list containing strings and probabilities. The cost of transmitting this list comes from two sources: the cost of transmitting the probabilities, and the cost of transmitting the strings. As a first approximation, let us assume the two are independent, a matter we shall return to in Section 2.

We begin by investigating the inherent cost/error tradeoff of transmitting a discrete probability distribution $\{p_j | 1 \leq j \leq k\}$ by uniform quantization to b bits. We divide the unit interval in $n = 2^b$ equal parts. For our theorems we will use a value b large enough so that we have $p_j \geq 2^{-(b-2)}$ for all j, leaving at least the first 4 bins empty. Usually 32 bits suffice for this, and as we shall see shortly, often a lot fewer are truly needed, though standard modeling tools like SRILM often use 64-bit quantities. For each probability, Alice sends b bits (the bin number). Bob, who knows b, reconstructs a value based on the center of the bin.

Since this process does not guarantee that the reconstructed values sum to 1, Bob takes the additional step of renormalizing these values: if $\sum q_i = r$, he will use $\overline{q} = q_i/r$ instead of the q_i that were transmitted by Alice. When b is large, the p_i will be distributed uniformly mod 2^{-b}. In this case, the expected values $E(p_i - q_i)$ are zero for all i, so $E(\sum q_i) = \sum E(q_i) = \sum E(p_i) = E(\sum p_i) = 1$ or, in other words, $E(r) = 1$. Since $\mathrm{Var}(r-1) = \sum_i \mathrm{Var}(p_i - q_i) = k/12n^2$ is on the order $1/n^2$, in the following estimate we can safely ignore the effects of renormalization. By Definition 2, the KL approximation error is

$$Q = \sum_{i=0}^{n-1} \sum_{i/n \leq p_j \leq (i+1)/n} p_j \Delta(p_j^i) \qquad (1)$$

where $\Delta(p_j^i) = \log(2np_j/(2i+1))$ is the difference between the logarithms of the actual p_j and the centerpoint of the interval $[i/n, (i+1)/n)$ where p_j falls. In absolute value, this is maximal when p_j is at the lower end of this interval, where $\Delta(p_j^i)$ is $\log(\frac{2i+1}{2i})$. Using the standard estimate $\log(1+x) \leq x$ this will be less than $\frac{1}{2i} \leq 1/8$ since $i \geq 4$. Since the $\Delta(p_j^i)$ are now estimated uniformly, and the p_j sum to 1, we obtain

Theorem 1. The approximation error Q_n of uniform quantization into n bins $[i/n, (i+1)/n)$ such that the first 4 bins are empty satisfies

$$Q_n \leq \frac{1}{8 \log 2} \sim 0.18 \qquad (2)$$

bits independent of n (the computation was in base e rather than base 2, hence the factor $\log 2$). With growing n the number of bins that remain empty will grow, and the estimate $\frac{1}{2i}$ of Δ can be improved accordingly.

Theorem 1 of course gives just an upper bound, and a rather crude one, the expected value of Q_n is considerably less. Instead of using the max value $\Delta(p_j^i)$ we can consider the expected absolute value, which is $\log(1+\frac{1}{2i})/n$, so equation (2) could be reformulated as

$$E(Q_n) \leq \frac{1}{8n \log 2} \qquad (3)$$

It is evident from the foregoing that the crux of the matter are the small p_i values, and at any rate, there can only be a handful of relatively large values, since the sum is 1. Experience shows that probabilities obtained from corpora span many orders of magnitude, which justifies the use of a log scale. Instead of the simple uniform quantization of Theorem 1, we will use a two-parameter quantization scheme, whereby first $\log p_j$ are cut off at $-C$, and the rest, which are on the $(-C, 0)$ interval, are sorted in $n = 2^b$ bins 'b-bit quantization'.

In effect, all probabilities below e^{-C} are assumed to be below measurement precision, and the log of the rest are uniformly quantized. We experimented with two simple techniques: representing the class $(-\infty, -C)$ with a very low fixed value (10^{-50}) or with one set to e^{-2C} based on the parameter C of the encoding. As there was no appreciable difference (which is not surprising given $\lim_{x \to 0} x \log x = 0$), from here on we simply speak of *zero weights* for weights below e^{-C}.

We emphasize that 'being below measurement precision' is not the same as 'being zero' in the above sense. First, in any corpus of size N the smallest number we can measure is $1/N$, yet we know that further strings that were not in our sample are not necessarily probability zero. It is therefore common to reserve a small fraction,

	∅	a	akár	bár	egyvala	más	másvala	minden	se	vala
hány	72383	9502	2432	55					21	4584
hogy	7781539	213687	3173	1839		4570		123	4138	31873
hol	117231	399052	1037	9845		16066		16009	20521	34081
honnan	24777	18628	296	1205		2482		1321	627	4274
honnét	1598	1197	12	25		78		33	23	236
hová	17589	21073	486	1753	1	5073	1	1859	2249	3966
hova	17360	10591	309	1166		1788		1381	2105	3036
ki	1309618	1464744	3933	60923	884		814	308508	165230	221175
meddig	11879	8171	189	225					74	252
mely	761277	1586913	166	74262	3				4	40601
melyik	68051	47564	1996	34477	2				939	48274
mennyi	76429	25805	657	1415					517	96184
mi	1626013	1303820	6500	52480	1337		161		275773	355690
miért	251120	20672	58	205	4				1810	13552
mikor	173652	555325	679	33516		15892		11288	206	18235
milyen	343643	38921	8217	68033		1618	1		55603	81155

Table 1: Frequencies of proquants in the Hungarian Webcorpus

generally 1-5% of the probability mass, to unseen events, and use calculated numbers, instead of measured values, to smooth the distribution. Unfortunately, the engineering philosophy behind the various backoff schemes (which often utilize MDL stopping criteria both in speech and language modeling, see e.g. Shinoda and Watanabe 2000, Seymore and Rosenfeld 1996) is diametrically opposed to the the method of inquiry preferred by linguists, whose primary interest is with generalization, i.e. with models that make falsifiable predictions, rather than furnishing descriptive statistics. In particular, negative generalizations, that something is forbidden by some rule of grammar, are just as interesting from their standpoint as positive generalizations. But how do we express a negative generalization?

Definition 5. A string will be deemed *ungrammatical* or *structurally excluded* iff every generation path includes at least one zero weight in the above sense.

If scores from different sources are multiplied together, the use of zero weights as markers of ungrammaticality is implicit in the semantics of WFSA.[1] Still, there are significant difficulties in implementing the idea. If we want to maintain the commonsensical assumption that **teh* is not a word (has zero unigram weight) and also account for the data that makes it the 34,174th most common string in English text, we will need to model typos. Once we learn that the log price of the /the/teh/ substitution is about -9.8, we can predict not just the frequency of *teh,* but also those of *weatehr, otehr, tehy, tehre, tehft,* and so forth, without adding these to the lexicon. Since such a model is based on computed frequencies of letter substitution and exchange rather than on the typos directly, the engineer has to give up the enterprise of building the entire language model in a single sweep directly on the data.

At the same time, the linguist has to give up the attractive simplicity of 'zero weight iff ungrammatical': the misspelling model will assign a low but nonzero weight to everything, and if this model is compiled together with a unigram model that contains only grammatical words, the simple world-view of Definition 5 will no longer work. Rather, we will have to say that it is zero weight in the *grammatical* subautomaton (visible only prior to getting compiled together with the semantic, spelling, stylistic, and possibly other subautomata) that defines grammaticality. We have to build an explicit noise model to make sense of the raw data, but this is not particularly surprising from the perspective of other sciences like astronomy where noise reduction is common practice.

The specific contribution of the MDL approach is that zero weights in the model are *a lot* cheaper than using low probabilities would be: the paradigm encourages both sparseness and struc-

[1] We owe this observation to an anonymous MOL referee.

tural decomposition. But before we can establish these points in Sections 2 and 3, we need to assimilate another piece of computational practice, the use of log probabilities. When quantization is uniform on the log scale, the expected value of the binning error is no longer zero, given our assumption of uniformity on the linear scale, but rather

$$\int_{-C\log\frac{i+1}{n}}^{-C\log\frac{i}{n}} e^x - e^{-C\frac{i+0.5}{n}}\,dx \sim C^3/8n^3 \qquad (4)$$

which yields an expected $r \sim kC^3/8n^3$, still negligible compared to the bound given in Theorem 2, which is obtained by methods similar to those used above.

Theorem 2. For C, n sufficiently large for the first 4 bins to remain empty, the approximation error L_n^C of log-uniform quantization with cutoff $-C$ into $n = 2^b$ bins $[-C(i+1)/n, -Ci/n)$ is bounded by

$$L_n^C \leq \frac{C}{2n\log 2} \qquad (5)$$

and the expected value $E(L_n^C)$ is bounded by $C^2/4n^2\log 2$.

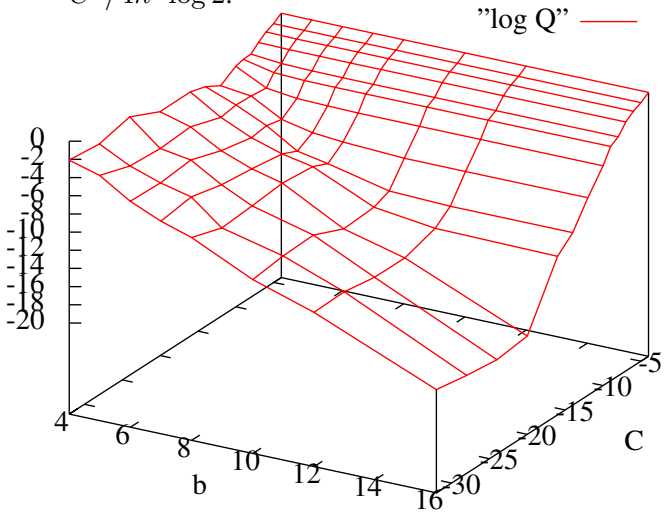

Figure 1: Error of log-scale uniform quantization

Let us see on an example how these error bounds compare to values obtained numerically. Our first example will explore what we will call, for want of a better name, the proquant system of Hungarian that covers both pro-forms (pronouns, proadjectives, proadverbials) and quantifiers. Given the prefixes *a-, minden-, vala-, egyvala-, másvala-, se-, akár-, más-, bár-* and zero, and suffixes *-ki, -mi, -hol, -hogy, -hova, etc.* we can create forms such as *valaki* 'someone', *valami* 'something', *akárki* 'anyone', *sehol* 'nowhere' and so on. Clearly, many of what we call prefixes and suffixes could be analyzed further, e.g. *másvala* as *más*+*vala*, but we don't want to prejudge the issue by presenting a maximally detailed analysis.

In a corpus of over 40 million sentences (Hungarian Webcorpus, Halácsy et al. 2004) we observed the frequencies in Table 1. Many of these proquant forms take inflectional suffixes (case, number, etc.), and the numbers presented here already include these, so that the 814 occurrences of *másvalaki* include forms like *másvalakivel* 'with someone else', *másvalakinek* 'to someone else' etc. If we think of the (stemmed) Hungarian vocabulary as a weighted language h, the set of prefixes (suffixes) as an unweighted language Pre (resp. Suff), the data is a sample from $S(h) \cap$ Pre·Suff with the weights renormalized. Altogether, we have 121 nonzero values plus 39 zeros, the entropy of the distribution is $H = 3.677$. Figure 1 plots the log of the observed quantization noise as a function of the number of bits b and the cutoff $-C$. Notice that once C is sufficiently large, no further gains are made by increasing it further. As expected from Theorem 2, the log of the error is roughly linear in $b = \log_2 n$ (the observed values are of course better than the bounds).

Definition 6. The *inherent noise* of a dataset D is the KL approximation error between a random subsample and its complement.

Ideally, we would want to compare another sample D' to D, but in many cases launching a comparable data collection effort is simply not feasible, and we must content ourselves with the simple procedure suggested by this definition. By randomly cutting the 40m sentence corpus on which the proquant dataset is based in 10m sentence parts and computing the KL divergence between any two, we obtain numbers in the 7-8·10^{-5} range, which means it makes little sense to approximate D with better precision than 10^{-5}. How to handle the singular cases when some q_j becomes 0 (as happens with half of the hapaxes when we cut the sample in two) is an issue we defer to Section 3.

Since the smallest p_j in this data is about 5·10^{-8}, by taking $C = 20$ we guarantee that no log probability is less than the cutoff point $-C$. Trivial 'list' automata consisting of an initial state,

a final state, and a separate Mealy arc (or Moore state) for each of the 121 nonzero observations already generate a weighted language within the inherent noise of the data at 10 bits, where the KL divergence is at $8 \cdot 10^{-6}$. At 12 bits, the divergence is below $1.4 \cdot 10^{-6}$, and at 16 bits, below $5 \cdot 10^{-9}$. As we shall see in the next Section, the MDL size of these models, between 2k and 7k bits, is dominated by factors unrelated to the precision b of the encoding.

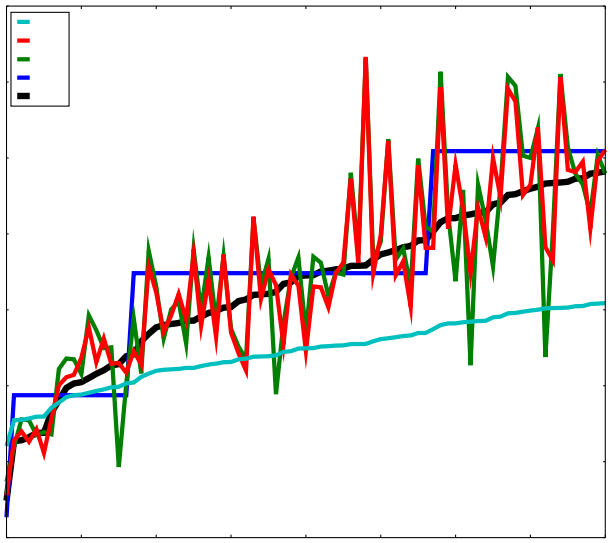

Figure 2: Model fit to observed probabilities

Figure 2 shows the 80 largest observed proquant probabilities (in black) in descending order, and the probabilities of the same strings as computed from several models. The 10 and 12-bit list automata are not plotted, as the computed values are graphically indistinguishable from the observed values, the rest will be discussed in the text.

2 Detecting ambiguity

Before turning to the actual MDL learning process, let us summarize what we have for the Hungarian proquant system so far. We have a weighted language of about 120 strings. When transmitting a weighted automaton, Alice is sending not strings and weights, but rather weight-labeled arcs and string-labeled states of a WFSA. In Definition 3 we used Moore machines, but in the literature Mealy automata, where inputs/outputs and weights are both tied to arcs are more common (see e.g. Mohri 2009). The rationale for preferring Moore over Mealy in the MDL context is that no gains can be obtained from joint compression of strings and probabilities (even though Mealy machines couple the two), while sharing of strings has very significant impact on MDL length, as we shall see shortly. For the simple 'list' automata this means adding extra states in the middle of a Mealy arc, and we need to take some precautions to guarantee that the representation is just as compact as it would be for a Mealy machine.

Let us now see in some detail how compact these encodings can get. With s states, and b bits for probability, an arc requires $2 \log_2 s + b$ bits. However, Bob can reasonably assume that Alice is only sending trimmed machines, with states that cannot be reached from the initial state or with no path to an accepting state already removed. Therefore, if Bob sees a state with no outbound path he supplies an outgoing arc, with probability 1, to the final state – such arcs need not be sent by Alice to begin with. Similarly, Bob can assume that all states except for the last one are non-accepting, and Alice will transmit information only to override this default when needed.

As for emissions, in a Moore machine each state emits a string (but no guarantees that different states emit different strings), so Alice needs to encode the strings somehow. If we assume that there is a character table shared between Alice and Bob, e.g. the character frequencies of Hungarian, with entropy H, encoding a string α costs simply $|\alpha|H$ bits. (We could take this also to be a case of transmitting a weighted language, but we assume that the cost of transmitting this language can be amortized over many WFSA that deal with Hungarian.)

b	l	M	c_s	c_a	KL	H_q
1	121	5210	4306	904	2.1883	6.833
2	121	5386	4306	1080	1.1207	3.487
3	121	5507	4306	1201	0.268	2.889
4	121	5628	4306	1322	0.041436	4.044
5	121	5749	4306	1443	0.016117	3.424
6	121	5870	4306	1564	0.002409	3.667
7	121	5991	4306	1685	0.000676	3.653
8	121	6112	4306	1806	0.000288	3.647
9	121	6233	4306	1927	5.905e-5	3.681
10	121	6354	4306	2048	8.003e-6	3.678
11	121	6475	4306	2169	3.999e-6	3.678
12	121	6596	4306	2290	1.387e-6	3.678
16	121	7080	4306	2774	4.660e-9	3.676

Table 2: List models with character-based string encoding

Table 2 summarizes the relevant values for the triv-

ial models where each weight gets its own trainable parameter. b is the number of bits, l is the number of trainable parameters (weights associated to arcs), c_a is the cost of transmitting the arcs. Note that this is less than $l(b+2\log_2 s)$, because of the redundancy assumptions shared by Alice and Bob. c_s is the cost of transmitting the emissions, and the total model cost is $M = c_a + c_s$. We would, ideally, also need to add to M a dozen bits or so to encode the major parameters of the coding scheme itself, such as the values $b = 10$ and $C = 20$, but these turn out to be negligible compared to the basic cost. Also, these major parameters are shared across the alternatives we compare, so whatever we do to minimize M will not be affected by uniformly adding (or uniformly ignoring) this constant cost. KL gives the KL divergence between the model and the training data. This measures the expected extra message length per arc weight, so that the error residual E is k times this value, where k is the number of values being modeled. We emphasize that $k = l$ only in the listing format, where all values are treated as independent – in the 'hub' model we shall discuss shortly l is only 26 (10 prefix and 16 suffix weights) but k is still 121.

The main components of the total MDL cost, M, $l \cdot KL$, $l(KL+H_q)$, and the total $M+l(KL+H_q)$ are plotted on Figure 3.

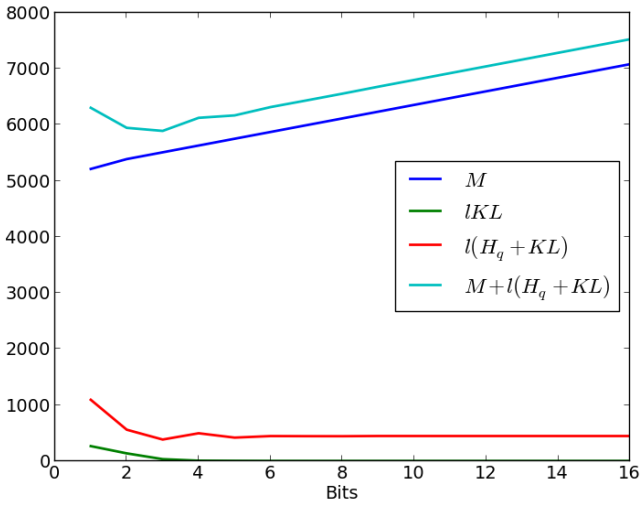

Figure 3: MDL cost components

All models with $b \geq 10$ are within the internal noise of the data, and it takes over 6kb to describe such a model. However, the bulk of these bits come from encoding the output strings character by character – if we assume that Alice and Bob share a morpheme table, the results improve a great deal, by over 3,600 bits. If the system recognizes what we already anticipated in Table 1, that each string can be expressed as the concatenation of a prefix and suffix, encoding the strings becomes drastically cheaper. Using MDL for segmentation is a well-explored area (see in particular Goldsmith 2001, Creutz and Lagus 2002, 2005), and we are satisfied by pointing out that using the morphemes in the first row and column of Table 1 we drastically reduce c_s, to about 708 bits, below the cost c_a of encoding the probabilities. The 3-bit list model providing the MDL optimum (dark blue in Figure 2) requires 1,900 bits with this string encoding, and is noticeably better than the SRILM bigram/trigram (turquoise) which takes around 12kb.

By encoding the emissions in a more clever fashion, we have not changed the structure of the model: the same states are still linked by the same arcs carrying the same probabilities, it is just the state labels that are now encoded differently. When expressed as a Mealy automaton, a listing of probabilities corresponds to a two-state WFSA with as many arcs as we have list elements (in our case, 121), while the arrangement of Table 1 is suggestive of a different model, one with 10 prefix arcs from the initial state to a central 'hub' state, and 16 suffix arcs from this hub to the final state.

We have trained such 'hub' models using KL, Euclidean (L_2), and absolute value (L_1) minimization techniques. Of these, direct minimization of KL divergence works best, obtaining 0.325 bits at $b = 10$, and 0.298 at $b = 12$ (red and green in Figure 2). While the difference, about 0.027 bits, is still perceptible compared to the noise level, with a signal to noise ratio (SNR) of 8 dB, it simply does not amortize over the 26 model probabilities we need to encode. Adding 2 bits for encoding one value requires a total of adding 52 bits to our specification of $\mathcal{M}$, while the gain of the error residual E, computed over the 121 observed values, is just 2.074 bits. In short, there is not much to be gained by going from 10 to 12 bits, and we need to look elsewhere for further compression.

Definition 7. For a weighted language p a model transform X is *learnable in principle* (LIP) if (i) both $\mathcal{M}$ and $X(\mathcal{M})$ are part of the hypothesis space and (ii) the total MDL cost of describing p by $X(\mathcal{M})$ is significantly below that of describing p by $\mathcal{M}$.

In a critical sense, LIP is weaker than MDL learn-

ability, since the space itself can be very large, and testing all hypothetical transforms X that fit the bill may not be feasible. The difference between LIP and practical MDL learnability is precisely the difference between existence proofs and constructive proofs. Our interest here is with the former: our goal is to demonstrate that structurally sound models are LIP. So far, we have seen that structurally valid segmentations can be effectively obtained by MDL. Our next task is to show that *ambiguity* is LIP.

As linguists, we know that the weakest point of the hub model is that *hogy*, accounting for almost 40% of the data, is not just a proquant 'how' but also a subordinating conjunction 'that'. To encode this ambiguity, we add another arc emitting *hogy* directly. Table 3 compares list models (lines 1-3, emissions encoded over morphemes rather than characters), simple hub models (lines 4-6), and hub models with this extra arc (lines 7-9).

b	l	M	c_s	c_a	KLe5	$M+E$
3	121	1907	705	1202	26800	2289
10	121	2754	705	2049	0.8	3199
12	121	2999	705	2290	0.14	3441
3	26	473	81	392	42343	1305
10	26	662	81	581	32593	1201
12	26	716	81	635	29827	1249
3	27	480	81	400	23094	1052
10	27	676	81	596	11268	1161
12	27	733	81	652	10022	1198

Table 3: Hub models with/out ambiguous *hogy*

As can be seen from the table, the best model again takes only 3 bits, but must include the extra parameter for handling the ambiguity of *hogy*. To learn this, at least in principle, without relying on the human knowledge that drove the heuristic search, consider the leading terms of the KL error. Arranging the $p_i \log(p_i/q_i)$ in order of decreasing absolute value we obtain *mi* 0.0192; *minden+ki* 0.0175; *a+mely* 0.0169; *mely* -0.0147; *a+mikor* 0.0135; *hogy* -0.0128; and so forth. Of all the 121 strings we may consider for direct emission, only *hogy* is worth adding a separate arc for. Further, if we repeat the process, adding a second direct arc never results in sufficient entropy gain compared to adding *hogy* alone.

To summarize, list models can approximate the original data within its inherent noise level, but incur a very significant MDL cost, even if they use an efficient string encoding because they keep many parameters, see the first three lines of Table 3 above. The hub models, which build structure similar to the one used in the string encoding, recognizing prefixes and suffixes for what they are, are far more compact, at 470-730 bits, even though they have a KL error of about .1-.4 bits. Finally, the hub+ambiguity model, with 27 parameters, reduces the total MDL cost to 1052 bits, less than half of the best list model.

Currently we lack the kind of detailed understanding of the description length surface over the WFSA$\times$stringencoding space that would let us say with absolute certainty that e.g. the hub model with ambiguous *hogy* is the global minimum, and we cannot muster the requisite computational power to exhaustively search the space of all WFSA with 27 arcs or less. Further gains could quite possibly made with even cruder quantization, e.g. to $n = 6$ levels (powers of 2 are convenient, but not essential for the model), or by bringing in non-uniform quantization.

On the one hand, we are virtually certain that the only encoding of emissions worth studying is the morpheme-based one, since the economy brought by this is tremendous, 3,600 bits over the proquants alone, and no doubt further gains elsewhere, as we extend the scope to other words that contain the same morphemes – in this regard, our findings simply confirm what Goldsmith, Creutz, Lagus, and others have already demonstrated. On the other hand, finding the right segmentation is only the first step, we also need a good model of the tactics. As we said at the beginning, the encoding of arcs and probabilities can to a significant extent be independent of the encoding of the emissions. Here the remarkable fact is that a better emission model could to a large extent drive the search for structuring the WFSA itself.

Given a segmentation of a string $\alpha = \alpha_1\alpha_2$, the hypothesis space includes both a single arc from some r to some t where we emit α, or the concatenation of two arcs $r \to s$ and $s \to t$ with s and t emitting α_1 and α_2 respectively. This brings in a bit of ambiguity in regards to the distribution of the probabilities, for if α had weight p the new arcs could be assigned any values p_1, p_2 as long as $p_1p_2 = p$, at least if the sum of outgoing probabilities from s remains 1. If s has no other arcs outgoing than $s \to t$ this forces $p_1 = p$, but if we collapse the intermediate states from several bimorphemic words, there is room for joint opti-

mization. In our example, collapsing all intermediate states in a single 'hub' halves the MDL cost.

3 Decomposition

For our next example we consider Hungarian stem-internal morphotactics. The Analytic Dictionary of Hungarian (Kiss et al 2011) provides, for each stem like *beleilleszt* 'fit in' an analysis like preverb+root+suffix wherein *bele* is one of a closed set of Hungarian preverbal particles, *ill* is the root, and *eszt* is a verb-forming suffix. There are six analytic categories: Stem S; sUffix U; Preverb P; root E; Modified M; and foreIgn I; so that each stem gets mapped on a string over $\Sigma = \{S, U, P, E, M, I\}$. We have two weighted languages: the *tYpe-weighted* language Y where each string is counted as many times as there are word types corresponding to it (so that e.g. for SUU we have 3,739 stems from *ábrándozik* 'daydream' to *zuhanyozó* 'shower stall', and the *tOken-weighted* language O where the same pattern has weight 18,739,068 because these words together appeared that many times in the Hungarian Webcorpus (Halácsy et al., 2004).

Since the inherent noise of O is about 0.0474 bits, we are interested in automata that approximate it within this limit. This is easily achieved with HMM-like WFSA that have arcs between any two states, using $b = 11$ bits or more, the smallest requiring only 781 bits. For Y the inherent noise is less, 0.011 bits, and the complete graph architecture, which only has 49 parameters (6 states, plus arcs from an initial state and arcs to a final state) is not capable of getting this close to the data, with the best models, from $b = 11$ onwards, remaining at KL distance 0.3. The two languages differ quite markedly in other respects as well, as can be seen from the fact that the character entropy of O is 0.933, that of Y is 1.567. Type frequency is not a good predictor of token frequency: the KL approximation error of O relative to Y is 2.11 bits.

An important aspect of the MDL calculus is the treatment of the singularities which arise whenever some of the q_i in Definition 2 are 0. In the case at hand, we find both types that are not attested in the corpus, and tokens whose type was not listed in the Analytic Dictionary, a situation that would theoretically render it impossible to compute the KL divergence in either direction. In practice, tokens with no dictionary type are either collected in a single 'unknown' type or are silently discarded. Both techniques have merit. The catch-all 'unknown' type can simply be assumed to follow the distribution of the known types, so a model that captures the data enshrined in the dictionary should, at least in principle, be also ideal for the data not seen by the lexicographer. Surprises may of course lurk in the unseen data, but as long as coverage is high, say P(unseen)≤ 0.05, surprises will really be restricted to this 5% of the unseen, or what is the same, will be at order P^2. In general we may consider two distributions $\{p_i\}$ and $\{q_i\}$ as in Definition 2, and compute $P = \sum_{q_i=0} p_i$, the proportion of q-singular data in p.

Theorem 3. The total cost L of transmitting an item from the p-distribution is bound by

$$L \leq (1-P)(KL(p,q)+H_q)+P(1+\log_2 n) \quad (6)$$

Proof We use, with probability $(1 - P)$, the q-based codebook: this will have cost H_q plus the modeling loss $KL(p, q)$. In the remaining cases (probability P) we should use a codebook based on the q-singular portion of p, but we resort to uniform coding at cost $\log_2 n$, where n is the number of singular cases. We need to transmit some information as to which codebook is used: this requires an extra $H(P, 1 - P) \leq P$ bits – collecting these terms gives (6). □

Theorem 3 gives a principled basis for discarding data within the MDL framework: when P is small, the second term of (6) can be absorbed in the noise. To give an example, consider the unigram frequencies listed in columns f_O and f_Y of Table 4. Some letters are quite rare, in particular, I makes up less than 0.06% of O and 0.013% of Y. Columns KL_O and KL_Y show the KL divergence of O and Y from models obtained by by discarding words containing the letter in question, columns P_O and P_Y show the weight of the strings that are getting discarded.

	f_O	P_O	KL_O	f_Y	P_Y	KL_Y
S	.7967	.9638	4.7887	.5342	.9122	3.5092
U	.1638	.1464	0.2284	.3443	.5699	1.2174
M	.0083	.0103	0.0149	.0255	.0623	0.0928
E	.0114	.0141	0.0205	.0331	.0804	0.1209
P	.0198	.0248	0.0362	.0623	.1531	0.2397
I	.0001	.0001	0.0002	.0006	.0010	0.0006

Table 4: Divergence caused by discarding data

In the case of Y, only I can be discarded while keeping below the inherent noise of the data, but for O we have three other symbols *M, E,* and *P*,

that could be removed. Further, removing both letters M, E only produces a KL loss of 0.036 bits; removing M, I a loss of 0.015 bits; E, I 0.021 bits; P, I 0.036 bits; and even removing all three of M, E, I only 0.036 bits.

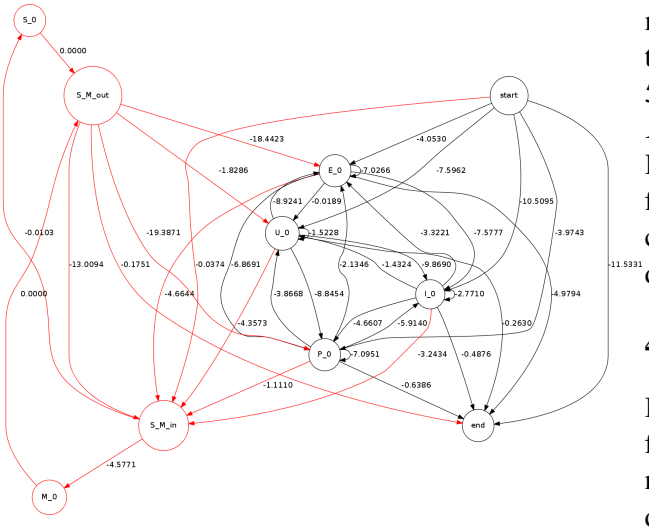

Figure 4: MS merge-split model

In the case of I, again as linguists we understand quite well what discarding this data means: we are excluding foreign stems. This is quite justified, not because foreign words like *paperback, pacemaker* or *baseball* are in any way inferior, but because their internal analysis is not transparent to the Hungarian reader (it is telling that the editors of the Analytic Dictionary coded the stem boundary in *paper+back* but not in *base+ball*).

Discarding M, a category that differs from S only in that the stem undergoes some automatic morphophonological change such as vowel elision, is also a sensible step in that the fundamental morphotactics are not at all affected by these changes, but how is this learnable, even in principle? Here we introduce another model transform called *XY merge-split* composed of two steps: first we replace all letters (or strings) X by Y and train a model, and next we split up the emission states of Y in the merged model to X and Y-emissions according to the relative proportions of X and Y in the original data.

For LIP, the key observation is that models constructed by *XY merge-split* have a transmission cost composed of two parts, the length of the smaller merged model (given in black in Figure 2), plus transmitting the pair X, Y and the probability of the split, which is exactly the cost of a single arc, even though the actual split model will have many more arcs (given in red in Figure 4). Once this is taken into account, we can systematically investigate all 6·5 merge-split possibilities. The results confirm the educated linguistic guess quite remarkably. The best compression rates are obtained by merging I with any of the minor categories or, if I is already discarded or merged in, merging M into S. The smallest O model before these steps took 781 bits, this is now reduced to 502 bits. If we start by discarding I, and merging M to S afterwards, this can be reduced to 349 bits. In the end we merge the morphophonologically affected forms with the ones not so affected not because our training as linguists tells us we should do this, but because that is what brevity demands.

4 Conclusions

In this paper we have developed an MDL-based framework for structure detection based on simple notions mostly borrowed from signal processing: quantization noise, inherent noise level, and cutoffs. Standard n-gram models fare rather poorly compared either in size or in model accuracy to the WFSA results obtained here: for example on the morphotactics data a straight SRILM trigram model has over 200 parameters and has KL divergence 1.09 bits. Most of the 64 bits per n-gram parameter are wasted (if we assume only 12 bits per parameter, the WFSA we use requires only 49 parameters and gets within 0.03 bits of the observed data) and further, the general-purpose backoff scheme built into SRILM just makes matters worse.

Similarly, on the proquant data an SRILM bigram model has 175 parameters (including the 26 unigram weights but excluding the backoff weights), yet it is farther from the data at 64 bits resolution than our best 27-parameter model at 3 bits. More important, the bigram structure of the proquant data has to be hand-fed into the standard model, while the MDL approach can discover this, together with other linguistically relevant observations such that *hogy* was ambiguous.

This is not to say that n-gram models are no longer competitive, for our current MDL methods, based on a simulated annealing learner, use too much CPU and will not scale to the gigaword regime without much further work. Yet if formal grammar and information theory are to get together again, as (Pereira, 2000) suggests, we must direct effort towards recapitulating linguistic practice, including the 'dirty' parts such as discarding data strategically. The main thrust of the work pre-

sented here is that the data manipulation methods that are the stock in trade of the descriptive linguist are LIP, and Universal Grammar is simply a short list of the permissible model transformations including path duplication for ambiguity, state merging for position class effects, and merge-split for collapsing categories.

Acknowledgments

We thank Dániel Varga (Prezi) and Viktor Nagy (Prezi) for the first version of the simulated annealing WFSA learner. Zséder wrote the version used in this study, Recski collected the data and ran the HMM baseline, Kornai advised. The current version benefited greatly from the remarks of anonymous referees. Work supported by OTKA grants #77476 and #82333.