Learning Pronunciation Dictionaries
Language Complexity and Word Selection Strategies

John Kominek Alan W Black
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jkominek,awb}@cs.cmu.edu

Abstract

The speed with which pronunciation dictionaries can be bootstrapped depends on the efficiency of learning algorithms and on the ordering of words presented to the user. This paper presents an active-learning word selection strategy that is mindful of human limitations. Learning rates approach that of an oracle system that knows the final LTS rule set.

1 Introduction

The construction of speech-to-speech translation systems is difficult, complex, and prohibitively expensive for all but handful of major languages. Developing systems for new languages is a highly skilled job requiring considerable effort, as is the process of training people to acquire the necessary technical knowledge.

Ideally, a native speaker of a (minor) language – with the right tools – should be able to develop a speech system with little or no technical knowledge of speech recognition, machine translation, dialog management, or speech synthesis. Rapid development of machine translation, for example, is the goal of (Lavie *et al*., 2003). Similarly, combined development of speech recognition and speech synthesis is the stated goal of (Engelbrecht and Schultz, 2005).

Here we concentrate on lexicon creation for synthesis and recognition tasks, with the affiliated problem of letter-to-sound rule inference. Two central questions of dictionary building are: what letter-to-sound rule representation lends itself well to incremental learning? – and which words should be presented to the user, in what order?

In this paper we investigate various approaches to the word ordering problem, including an active learning algorithm. An "active learner" is a class of machine learning algorithms that choose the order in which it is exposed to training examples (Auer, 2000). This is valuable when there isn't a pre-existing set of training data and when the cost of acquiring such data is high. When humans are adding dictionary entries the time and accuracy depends on the selected word (short words are easier than long; familiar are easier than unfamiliar), and on how quickly the learner's error rate drops (long words are more informative than short). Also, mindful that no answer key exists for new languages – and that humans easily become impatient – we would like to know when a language's letter to sound rule system is, say, 90% complete. This turns out to be surprising elusive to pin down.

The next section outlines our working assumptions and issues we seek to address. Section 3 describes our LTS learning framework, an elaboration of (Davel and Barnard, 2003). The learning behavior on multiple test languages is documented in Section 4, followed in Section 5 by a comparison of several word selection strategies.

2 Assumptions and Issues

In designing language technology development tools we find it helpful to envision our target user, whom may be characterized as "non-technical." Such a person speaks, reads, and writes the target language, is able to enumerate the character set of that language, distinguish punctuation from whitespace, numerals, and regular letters or graphemes, and specify if the language distinguishes upper and lower casing. When presented

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 232–239,
New York, June 2006.

with the pronunciation of a word (as a synthesized wavefile), the user can say whether it is right or wrong. In addition, such a person has basic computer fluency, can record sound files, and can navigate the HTML interface of our software tools. If these latter requirements present a barrier then we assume the availability of a field agent to configure the computer, familiarize the user, plus translate the English instructions, if necessary.

Ideally, our target user need not have explicit knowledge of their own language's phoneme set, nor even be aware that a word can be transcribed as a sequence of phonemes (differently from letters). The ability to reliably discover a workable phoneme set from an unlabeled corpus of speech is not yet at hand, however. Instead we elicit a language's phoneme set during an initialization stage by presenting examples of IPA wavefiles (Wells and House, 1995).

Currently, pronunciations are spelled out using a romanized phonetic alphabet. Following the recommendation of (Davel and Barnard, 2005) a candidate pronunciation is accompanied with a wavefile generated from a phoneme-concatenation synthesizer. Where possible, more than one pronunciation is generated for each word presented, under that assumption that it is easier for a listener to select from among a small number of choices than correct a wrong prediction.

2.1 Four Questions to Address

1. *What is our measure of success?* Ultimately, the time to build a lexicon of a certain coverage and correctness. As a proxy for time we use the number of characters presented. (Not words, as is typically the case, since long words contain more information than short, and yet are harder for a human to verify.)
2. *For a given language, how many words (letters) are needed to learn its LTS rule system?* The true, yet not too useful answer is "it depends." The complexity of the relation between graphemic representation and acoustic realization varies greatly across languages. That being the case, we seek a useful measure of a language's degree of complexity.
3. *Can the asymptote of the LTS system be estimated, so that one can determine when the learned rules are 90 or 95% complete?* In Section 4 we present evidence that this may not be possible. The fall-back position is percentage coverage of the supplied corpus.
4. *Which words should be presented to the user, and in what order?* Each additional word should maximize the marginal information gain to the system. However, short words are easier for humans to contend with than long. Thus a length-based weighting needs to be considered.

3 LTS Algorithm Basics

A wide variety of approaches have been applied to the problem of letter-to-sound rule induction. Due to simplicity of representation and ease of manipulation, our LTS rule learner follows the Default & Refine algorithm of Davel (Davel and Barnard, 2004). In this framework, each letter *c* is assigned a default production p_1-p_2... denoting the sequence of zero or more phonemes most often associated with that letter. Any exceptions to a letter's default rule is explained in terms of the surrounding context of letters. The default rules have a context width of one (the letter itself), while each additional letter increases the width of the context window. For example, if we are considering the first occurrence of 's' in the word *basics,* the context windows are as listed in Table 1. By convention, the underscore character denotes the predicted position, while the hash represents word termination.

width	*context sets ordered by increasing width*
1	{_}
2	{a_ , _i}
3	{ba_ , a_i , _ic}
4	(#ba_ , ba_i , a_ic , _ics}
5	{#ba_i , ba_ic , a_ics , _ics#}
6	{#ba_ic , ba_ics , a_ics#}
7	{#ba_ics , ba_ics#}
8	{#ba_ics#}

Table 1. Letter contexts for the first 's' in *basics*.

In this position there are 20 possible explanatory contexts. The order in which they are visited defines an algorithm's search strategy. In the class of algorithms knows as "dynamically expanding context (DEC)", contexts are considered top-down as depicted in Table 1. Within one row, some algorithms follow a fixed order (e.g. center, left, right). Another variant tallies the instances of productions

associated with a candidate context and chooses the one with the largest count. For example, in Spanish the letter 'c' may generate K (65%), or TH when followed by e or i (32%), or CH when followed by h (3%). These are organized by frequency into a "rule chain."

Rule rank	*RHS*	*Context*	*Frequency*
1	K	_	65.1%
2	TH	_i	23.6%
3	TH	_e	8.5%
4	CH	_h	2.8%

If desired, rules 2 and 3 in this example can be condensed into 'c' → TH /_{i,e}, but in general are left separated for sake of simplicity.

In our variant, before adding a new rule all possible contexts of all lengths are considered when selecting the best one. Thus the rule chains do not obey a strict order of expanding windows, though shorter contexts generally precede longer ones in the rule chains.

One limitation of our representation is that it does not support gaps in the letter context. Consider the word pairs tom/tome, top/tope, tot/tote. A CART tree can represent this pattern with the rule: *if* (c_{-1} = 't' *and* c_0='o' *and* c_2='e') *then ph=OW*. In practice, the inability to skip letters is not a handicap.

3.1 Multiple Pronunciation Predictions

Given a word, finding the predicted pronunciation is easy. Rule chains are indexed by the letter to be predicted, and possible contexts are scanned starting from the most specific until a match is found. Continuing our example, the first letter in the Spanish word *ciento* fails rule 4, fails rule 3, then matches rule 2 to yield TH. For additional pronunciations the search continues until another match is found: here, the default rule 'c' → K /_. This procedure is akin to predicting from progressively smoother models. In a complex language such as English, a ten letter word can readily generate dozens of alternate pronunciations, necessitating an ordering policy to keep the total manageable.

4 Language Characterization

English is notorious for having a highly irregular spelling system. Conversely, Spanish is admired for its simplicity. Most others lie somewhere in between. To estimate how many words need to be seen in order to acquire 90% coverage of a language's LTS rules, it helps to have a quantitative measure. In this section we offer a perplexity-based measure of LTS regularity and present measurements of several languages with varying corpus size. These measurements establish, surprisingly, that a rule system's perplexity increases without bound as the number of training words increases. This holds true whether the language is simple or complex. In response, we resort to a heuristic measure for positioning languages on a scale of relative difficulty.

4.1 A Test Suite of Seven Languages

Our test suite consists of pronunciation dictionaries from seven languages, with English considered under two manifestations.

English. Version 0.6d of CMU-DICT, considered without stress (39 phones) and with two level stress marking (58 phones). ***German***. The Celex dictionary of 321k entries (Burnage, 1990). ***Dutch***. The Fonilex dictionary of 218k entries (Mertens and Vercammen, 1998). Fonilex defines an abstract phonological level from which specific dialects are specified. We tested on the "standard" dialect. ***Afrikaans***. A 37k dictionary developed locally. Afrikaans is a language of South Africa and is a recent derivative of Dutch. ***Italian***. A 410k dictionary distributed as part of a free Festival-based Italian synthesizer (Cosi, 2000). ***Spanish***. Generated by applying a set of hand written rules to a 52k lexicon. The LTS rules are a part of the standard Festival Spanish distribution. ***Telugu***. An 8k locally developed dictionary. In its native orthography, this language of India possess a highly regular syllabic writing system. We've adopted a version of the Itrans-3 transliteration scheme (Kishore 2003) in which sequences of two to four English letters map onto Telugu phonemes.

4.2 Perplexity as a Measure of Difficulty

A useful way of considering letter to sound production is as a Markov process in which the generator passes through a sequence of states (letters), each probabilistically emitting observation symbols (phonemes) before transitioning to the next state (following letter). For a letter c, the unpredictability of phoneme emission is its entropy $H(c)=-\sum(p_i \log p_i)$ or equivalently its perplexity $P(c)=e^{H(c)}$. The perplexity can be interpreted as

the average number of output symbols generated by a letter. The production perplexity of the character set is the sum of each individual letter's perplexity weighted by its unigram probability p_c.

$$Per_{ave} = \sum_c p_c e^{-\sum_i p_i \log p_i} \quad (1)$$

Continuing with our Spanish example, the letter 'c' emits the observation symbols (K, TH, CH) with a probability distribution of (.651, .321, .028), for a perplexity of 2.105. This computation applies when each letter is assigned a single probabilistic state. The process of LTS rule discovery effectively splits the state 'c' into four context-defined substates: (-,c,-), (-,c,i), (-,c,e), (-,c,h). Each of these states emits only a single symbol. Rule addition is therefore an entropy reduction process; when the rule set is complete the letter-to-sound system has a perplexity of 1, i.e. it is perfectly predictable.

The "price paid" for perfect predictability is a complex set of rule chains. To measure rule complexity we again associate a single state with each letter. But, instead of phonemes, the *rules* are the emission symbols. Thus the letter 'c' emits the symbols (K/_, TH/_i, TH/_e, CH/_h) with a distribution of (.651, .236, .085, .028), for a perplexity of 2.534. Applying equation (1) to the full set of rules defines the LTS system's average perplexity.

4.3 Empirical Measurements

In the Default & Refine representation, the rule chain for each letter is is initialized with its most probably production. Additional context-dependent rules are appended to cover additional letter productions, with the rule offering the greatest incremental coverage being added first. (Ties are broken in an implementation-dependent way.)

Figure 1 uses Spanish to illustrate a characteristic pattern: the increase in coverage as rules are added one at a time. Since the figure of merit is letter-based, the upper curve (% letters correct) increases monotonically, while the middle curve (% words correct) can plateau or decrease briefly.

In the lower curve of Figure 1 the growth procedure is constrained such that all width 1 rules are added before width 2 rules, which in turn must be exhausted before width 3 rules are considered. This constraint leads to its distinctive scalloped shape. The upper limit of the W=1 region shows the performance of the unaided default rules (68% words correct).

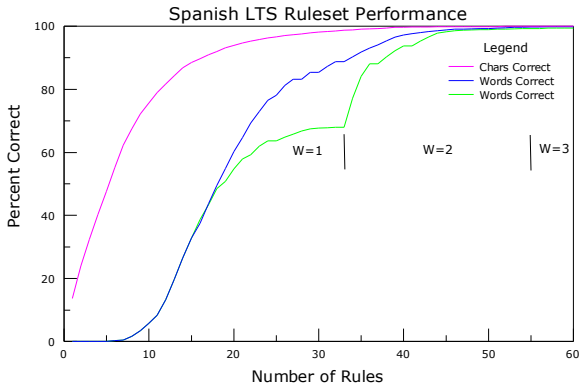

Figure 1. Coverage of Spanish (52k corpus) as a function of rule size. For the lower curve, W indicates the rule context window width. The middle (blue) curve tracks near-optimal performance improvement with the introduction of new rules.

For more complex languages the majority of rules have a context width in the range of 3 to 6. This is seen in Figure 2 for English, Dutch, Afrikaans, and Italian. However, a larger rule set does not mean that the average context width is greater. In Table 2, below, compare Italian to Dutch.

Language	*Number of Rules*	*Average Width*
English 40k	19231	5.06
Dutch 40k	10071	4.35
Afrikaans 37k	5993	4.66
Italian 40k	3385	4.78
Spanish 52k	76	1.66

Table 2. Number of LTS rules for five language and their average context width.

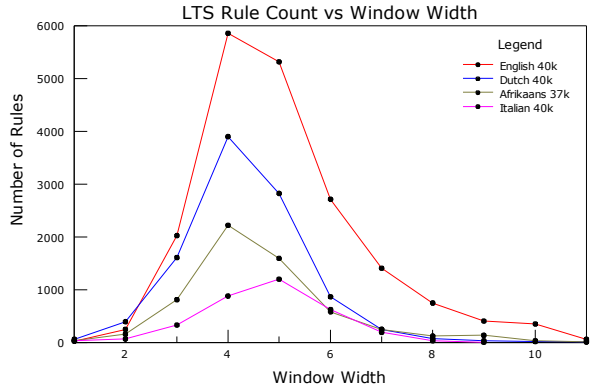

Figure 2. Distribution of LTS rules by context window width for four languages: English, Dutch, Afrikaans, and Italian.

Beyond a window width of 7, rule growth tapers off considerably. In this region most new rules serve to identify particular words of irregular spelling, as it is uncommon for long rules to generalize beyond a single instance. Thus when training a smoothed LTS rule system it is fair to ignore contexts larger than 7, as is done for example in the Festival synthesis system (Black, 1998).

Figure 2 contrasts four languages with training data of around 40k words, but says nothing of how rule sets grow as the corpus size increases. Figure 3 summarizes measurements taken on eight encodings of seven languages (English twice, with and without stress marking), tested from a range of 100 words to over 100,000. Words were subsampled from each alphabetized lexicon at equal spacings. The results are interesting, and for us, unexpected.

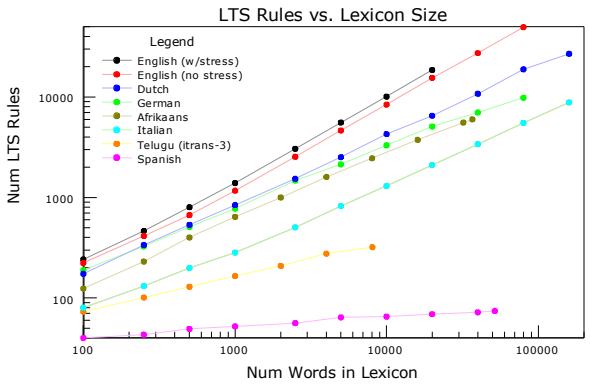

Figure 3. Rule system growth as the corpus size is increased, for seven languages. From top to bottom: English (twice), Dutch, German, Afrikaans, Italian, Telugu, Spanish. The Telugu lexicon uses an Itrans-3 encoding into roman characters, not the native script, which is a nearly perfect syllabic transcription. The context window has a maximum width of 9 in these experiments.

Within this experimental range none of the languages reach an asymptotic limit, though some hint at slowed growth near the upper end. A straight line on a log-log graph is characteristic of geometric growth, to which a power law function $y=ax^b+c$ is an appropriate parametric fit. For difficult languages the growth rates (power exponent b) vary between 0.5 and 0.9, as summarized in Table 3. The language with the fastest growth is English, followed, not by Dutch, but Italian. Italian is nonetheless the simpler of these two, as indicated by the smaller multiplicative factor a.

Language	***a***	***b***
English (stressed)	2.97	0.88
English (plain)	3.27	0.85
Dutch	12.6	0.64
German	39.86	0.49
Afrikaans	15.34	0.57
Italian	2.16	0.69

Table 3. Parameters a and b for the power law fit $y=ax^b+c$ to the growth of LTS system size.

It would be good if a tight ceiling could be estimated from partial data in order to know (and report to the lexicon builder) that with n rules defined the system is m percent complete. However, this trend of geometric growth suggests that asking "how many letter-to-sound rules does a given language have?" is an ill-posed question.

In light of this, two questions are worth asking. First, is the geometric trend particular to our rule representation? And second, is "total number of rules" the right measure of LTS complexity? To answer the first question we repeated the experiments with the CART tree builder available from the Festival speech synthesis toolkit. As it turns out – see Table 4 – a comparison of contextual rules and node counts for Italian demonstrate that a CART tree representation also exhibits geometric growth with respect to lexicon size.

Num Words in Lexicon	***Contextual LTS Rules***	***CART Tree Nodes***
100	80	145
250	131	272
500	198	399
1000	283	601
2500	506	1169
5000	821	1888
10,000	1306	2840
20,000	2109	4642
40,000	3385	7582
80,000	5524	13206

Table 4. A comparison of rule system growth for Italian as the corpus size is increased. CART tree nodes (i.e. questions) are the element comparable to LTS rules used in letter context chains. The fitted parameters to the CART data are a=2.29 and b=0.765. This compares to a=2.16 and b=0.69.

If geometric growth and lack of an obvious asymptote is not particular to expanding context rule chains, then what of the measure? The measure proposed in Section 4.2 is average chain perplexity. The hypothesis is that a system close to saturation will still add new rules, but that the average perplexity levels off. Instead, the data shows little sign of saturation (Figure 4). In contrast, the average perplexity of the letter-to-phoneme distributions remains level with corpus size (Figure 5).

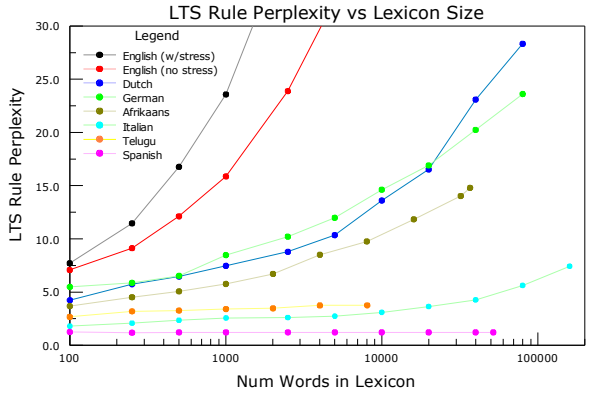

Figure 4. Growth of average rule perplexity as a function of lexicon size. Except for Spanish and Telugu, the average rule system perplexity not only grows, but grows at an accelerating rate.

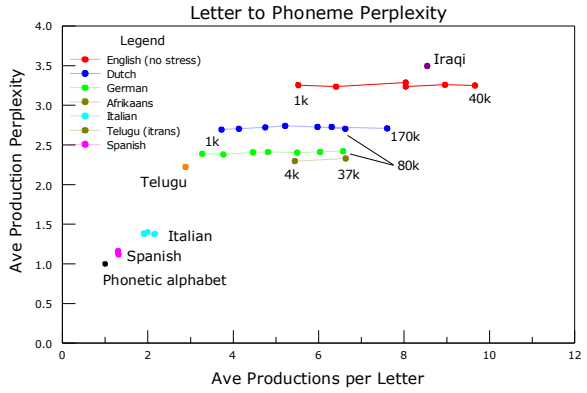

Figure 5. Growth of average letter-to-phoneme production perplexity as a function of lexicon size.

Considering these observations we've resorted to the following heuristic to measure language complexity: a) fix the window width to 5, b) measure the average rule perplexity at lexicon sizes of 10k, 20k, and 40k, then c) take the average of these three values. Fixing the window width to 5 is somewhat arbitrary, but is intended to prevent the system from learning an unbounded suite of exceptions. Available values are contained in Table 5.

Language	***Ave Letter Perplexity***	***Heuristic Perplexity***	***Perplexity Ratio***
English	3.25	50.11	15.42
Dutch	2.73	16.80	6.15
German	2.41	16.70	6.93
Afrikaans	2.32	11.48	8.32
Italian	1.38	3.52	2.55
Spanish	1.16	1.21	1.04

Table 5. Perplexity measures for six languages. The third (rightmost) column is the ratio of the second divided by the first. A purely phonetic system has a heuristic perplexity of one.

From these measurements we conclude, for example, that Dutch and German are equally difficult, that English is 3 times more complex than either of these, and that English is 40 times more complex than Spanish.

5 Word Selection Strategies

A selection strategy is a method for choosing an ordered list of words from a lexicon. It may be based on an estimate of expected maximum return, or be as simple as random selection. A good strategy should enable rapid learning, avoid repetition, be robust, and not overtax the human verifier.

This section compares competing selection strategies on a single lexicon. We've chosen a 10k Italian lexicon as a problem of intermediate difficulty, and focus on early stage learning. To provide a useful frame of reference, Figure 6 shows the results of running 5000 experiments in which the word sequence has been chosen randomly. The x-axis is number of letters examined.

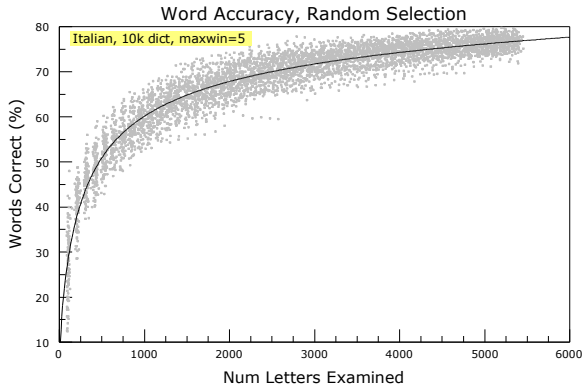

Figure 6. Random sampling of Italian 10k corpus.

Figure 7 compares average random performance to four deterministic strategies. They are: alphabetical word ordering, reverse alphabetical, alphabetical sorted by word length (groups of single character words first, followed by two character words, etc.), and a greedy ngram search. Of the first three, reverse alphabetical performs best because it introduces a greater variety of ngrams more quickly than the others. Yet, all of these three are substantially worse than random. Notice that grouping words from short to long degrades performance. This implies that strategies tuned to the needs of humans will incur a machine learning penalty.

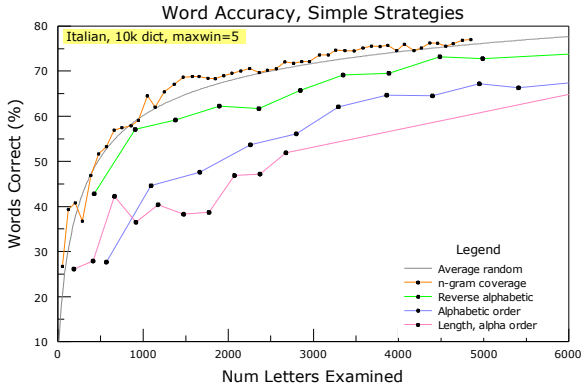

Figure 7. Comparison of three simple word orderings to the average random curve, as well as greedy ngram search.

It might be expected that selecting words containing the most popular ngrams first would out-performs random, but as is seen in Figure 7, greedy selection closely tracks the random curve. This leads us to investigate active leaning algorithms, which we treat as variants of ngram selection.

5.1 Algorithm Description

Let W = $\{w_1,w_2,...\}$ be the lexicon word set, having A = {'a', 'b',...} as the alphabet of letters. We seek an ordered list V = (... w_i ...) s.t. score(w_i) ≥ score (w_{i+1}). V is initially empty and is extended one word at a time with w_b, the "best" new word. Let $g=c_1c_2...c_n \in A^*$ be an ngram of length n, and $G_w=\{g_i\}$, $g_i \in w$ are all the ngrams found in word w. Then $G_W = \uplus G_w$, $w \in W$, is the set of all ngrams in the lexicon W, and $G_V = \uplus G_w$, $w \in V$ is the set of all ngrams in the selected word list V. The number of occurrences of g in W is score(g), while score(w) = ∑ score(g) st. $g \in w$ and $g \notin G_V$. The scored ngrams are segmented into separately sorted lists, forming an ordered list of queues Q = $(q_1,q_2,...q_N)$ where q_n contains ngram of length n and only n.

Algorithm

```
for q in Q
    g = pop(q)
    for L = 1 to |longest word in W|
        W_{g,L} = {w_i} s.t. |w_i| = L, g ∈ w_i and w_i ∉ V
        w_b = argmax score(W_{g,L})
        if score (w_b) > 0 then
            V = V + w_b
            G_V = G_V ∪ G_wb
            return w_b
```

In this search the outer loop orders ngrams by length, while the inner loop orders words by length. For selection based on ngram coverage, the queue Q is computed only once for the given lexicon W. In our active learner, Q is re-evaluated after each word is selected, based on the ngrams present in the current LTS rule contexts. Let $G_{LTS} = \{g_i\}$ s.t. $g_i \in$ some letter context in the LTS rules. Initially $G_{LTS,0} = \{\}$. Then, at any iteration k, $G_{LTS,k}$ are the ngrams present in the rules, and $G'_{LTS,k+1}$ is an expanded set of candidate ngrams that constitute the elements of Q. G' is formed by prepending each letter c of A to each g in G, plus appending each c to g. That is, $G'_{LTS,k+1} = A \times G_{LTS,k} \cup G_{LTS,k} \times A$ where × is the Cartesian product. Executing the algorithm returns w_b and yields $G_{LTS,k+1}$ the set of ngrams covered by the expanded rule set. In this way knowledge of the current LTS rules guides the search for maximally informative new words.

5.2 Active Learner Performance

Figure 8 displays the performance of our active learner on the Italian 10k corpus, shown as the blue curve. For the first 500 characters encountered, the active learner's performance is almost everywhere better than average random, typically one half to one standard deviation above this reference level.

Two other references are shown. Immediately above the active learner curve is "Oracle" word selection. The Oracle has access to the final LTS system and selects words that maximally increases coverage of the known rules. The topmost curve is for a "Perfect Oracle." This represents an even more unrealistic situation in which each letter of each word carries with it information about the corresponding production rule. For example, that 'g' yields /F/ 10% of the time, when followed by the letter 'h' (as in "laugh") . Carrying complete information with each letter allows the LTS system to be constructed directly and without mistake. In contrast, the non-perfect oracle makes mistakes sequencing rules in each letter's rule chain. This decreases performance.

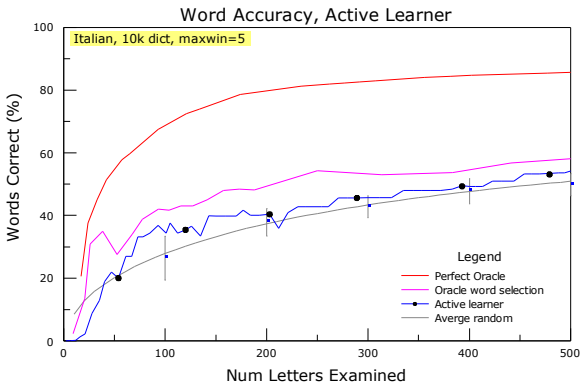

Figure 8. From top to bottom: a perfect Oracle, a word selection Oracle, our active learner, and average random performance. The perfect Oracle demarcates (impossibly high) optimal performance, while Oracle word selection suggests near-optimality. For comparison, standard deviation error bars are added to the random curve.

Encouragingly, the active learning algorithm straddles the range in between average random (the baseline) and Oracle word selection (near-optimality). Less favorable is the non-monotonicity of the performance curve; for example, when the number of letters examined is 135, and 210. Analysis shows that these drops occur when a new letter-to-sound production is encountered but more than one context offers an equally likely explanation. Faced with a tie, the LTS learner sometimes chooses incorrectly. Not being aware of this mistake it does not seek out correcting words. Flat plateaus occur when additional words (containing the next most popular ngrams) do not contain previously unseen letter-to-sound productions.

6 Conclusions

While this work does not definitively answer the question of "how may words to learn the rules," we have developed ways of characterizing language complexity, which can guide developers. We've devised a word selection strategy that appears to perform better than the (surprisingly high) standard set by randomly selection. Further improvements are possible by incorporating knowledge of word alignment and rule sequencing errors. By design, our strategy is biased towards short words over long, thereby being "nice" to lexicon developers – our original objective.

Acknowledgments

This material is in part based upon work supported by the National Science Foundation under Grant No. 0415201. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Peter Auer, 2000. *Using upper confidence bounds for online learning*. Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp.

Alan W Black, Kevin Lenzo, and Vincent Pagel, 1998. *Issues in Building General Letter to Sound Rules.* 3rd ESCA Workshop on Speech Synthesis, Australia.

Gavin Burnage, 1990. *CELEX – A Guide for Users*. Hijmegen: Centre for Lexical Information, University of Nijmegen.

Piero Cosi, Roberto Gretter, Fabio Tesser, 2000. *Festival parla italiano*. Proceedings of GFS2000, Giornate del Gruppo di Fonetica Sperimentale, Padova.

Marelie Davel and Etienne Barnard, 2003. *Bootstrapping in Language Resource Generation.* Proceedings of the 14th Symposium of the Pattern Recognition Association of South Africa, pp. 97-100.

Marelie Davel and Etienne Barnard, 2004. *A default-and-refine approach to pronunciation prediction*, Proceedings of the 15th Symposium of the Pattern Recognition Association of South Africa.

Marelie Davel and Etienne Barnard, 2005. *Bootstrapping Pronunciation Dictionaries: Practical Issues.* Proceedings of the 9th International Conference on Spoken Language Processing, Lisbon, Portugal.

Herman Engelbrecht, Tanja Schultz, 2005. *Rapid Development of an Afrikaans-English Speech-to-Speech Translator*, International Workshop on Spoken Language Translation, Pittsburgh, PA. pp.169-176.

S P Kishore and Alan W Black, 2003. *Unit Size in Unit Selection Speech Synthesis.* Proceedings of the 8th European Conference on Spoken Language Processing, Geneva, Switzerland.

Alon Lavie, et al. 2003. *Experiments with a Hindi-to-English Transfer-based MT System under a Miserly Data Scenario*, ACM Transactions on Asian Language Information Processing, 2(2).

Piet Mertens and Filip Vercammen, 1998. *Fonilex Manual*, Technical Report, K. U. Leuven CCL.

John Wells and Jill House, 1995. *Sounds of the IPA*. http://www.phon.ucl.ac.uk/shop/soundsipa.php.