Bootstrapping pronunciation models

M. Davel* and E. Barnard*

Bootstrapping techniques can accelerate the development of language technology for resource-scarce languages. We define a framework for the analysis of a general bootstrapping process whereby a model is improved through a controlled series of increments, at each stage using the previous model to generate the next. We apply this framework to the task of creating pronunciation models for resource-scarce languages, iteratively combining machine learning and human knowledge in a way that minimizes the human intervention required during this process. We analyse the effectiveness of such an approach when developing a medium-sized (5000–10 000 word) pronunciation lexicon. We develop such an electronic pronunciation lexicon in Afrikaans, one of South Africa's official languages, and provide initial results obtained for similar lexicons developed in Zulu and Sepedi, two other South African languages. We derive a mathematical model that can be used to predict the amount of time required for the development of a pronunciation lexicon of a given size, demonstrate the various tools that can accelerate the bootstrapping process, and evaluate the efficiency of these tools in practice.

Introduction

Human language technologies (HLT) hold much promise for the developing world, especially for user communities that have a low literacy rate, speak a minority language, or reside in areas where access to conventional information infrastructure is limited. For example, information systems that provide speech-enabled services via a telephone can serve a user in his or her language of choice in a remote location without requiring additional expertise from the user or a sophisticated Internet infrastructure. In South Africa, while Internet penetration is still low, more than 90% of the population has access to a telephone and the potential of voice services for improving access to information is receiving increasing attention.[1]

The development of various forms of HLT — such as speech-based information systems, speech recognition, speech synthesis or multilingual information retrieval systems — requires the availability of extensive language resources. The development of these resources involves significant effort and can be a prohibitively expensive task when such technologies are developed for a resource-scarce language. This presents a significant obstacle to the development of HLT in the developing world, where few electronic resources are available for local languages, skilled computational linguists are scarce and linguistic diversity is high.

One such language resource required by many speech processing tasks is a pronunciation model. A pronunciation model for a specific language describes the process of letter-to-sound conversion: given the orthography of a word, it provides a prediction of the phonemic realization of that word. This component is required by many speech processing tasks — including general domain speech synthesis and large vocabulary speech recognition — and is often one of the first resources required when developing speech technology in a resource-scarce language.

In order to accelerate the uptake of HLT in the developing world, techniques are required that support the efficient development of language resources. In prior work,[2–4] we have developed bootstrapping techniques to accelerate the development of a pronunciation model in a resource-scarce language. During the procedure known as 'bootstrapping', a model is improved iteratively via a controlled series of increments, at each stage using the previous model to generate the next. This self-improving circularity distinguishes bootstrapping from other incremental learning processes and is the key to its efficiency, as we discuss below.

In this work we define a generic framework for analysing a bootstrapping process and apply this framework to the task of creating pronunciation models for resource-scarce languages. We use the framework to derive a mathematical model that can be used to predict the amount of time required for the development of a pronunciation lexicon of a given size, analyse the effectiveness of the approach when developing a medium-sized (5 000–10 000 word) pronunciation lexicon and demonstrate the various tools that can accelerate the bootstrapping process.

The paper is structured as follows: in the following section we provide background with regard to the bootstrapping of pronunciation models, and then we define a generic framework for analysing a bootstrapping process. We next describe the specific bootstrapping approach followed during the development of an Afrikaans pronunciation lexicon, analyse the results achieved and evaluate the efficiency of the process. Finally, we discuss initial results obtained when bootstrapping pronunciation models in two additional languages, Zulu and Sepedi, and discuss the implications of our results.

Background

In prior work, we developed an audio-enabled approach to the bootstrapping of pronunciation models, and demonstrated the efficiency of this process for small lexicons.[2,3] The core of this bootstrapping process relies on an efficient grapheme-to-phoneme pronunciation prediction algorithm. This algorithm is used to generalize from the existing lexicon (dictionary) in order to predict additional entries, increasing the size of the lexicon in an incremental fashion. The bootstrapping system is initialized with a large word list (containing no pronunciation information), or with a pre-existing pronunciation lexicon if such a resource is available. The system chooses the next 'best' word to consider, predicts a pronunciation for this word and presents a human dictionary developer with an audio version[a] of the predicted pronunciation. The human acts as a 'verifier' and provides a verdict with regard to the accuracy of the word-pronunciation pair: whether the pronunciation is correct as predicted, or not. The verifier can also indicate that the word itself is invalid (e.g. from a different language or a misspelled word), ambiguous depending on context (e.g. the word 'read'

*HLT Research Group, CSIR Meraka Institute, P.O. Box 395, Pretoria 0001, South Africa.
E-mail: mdavel@csir.co.za; ebarnard@csir.co.za

[a]The audio version is created from the predicted phoneme string by concatenating audio samples of the specified phonemes, that is, the word is 'sounded' rather than synthesized. This implies that the only additional resource requirement is a set of sample phonemes, rather than a more complex speech synthesizer.

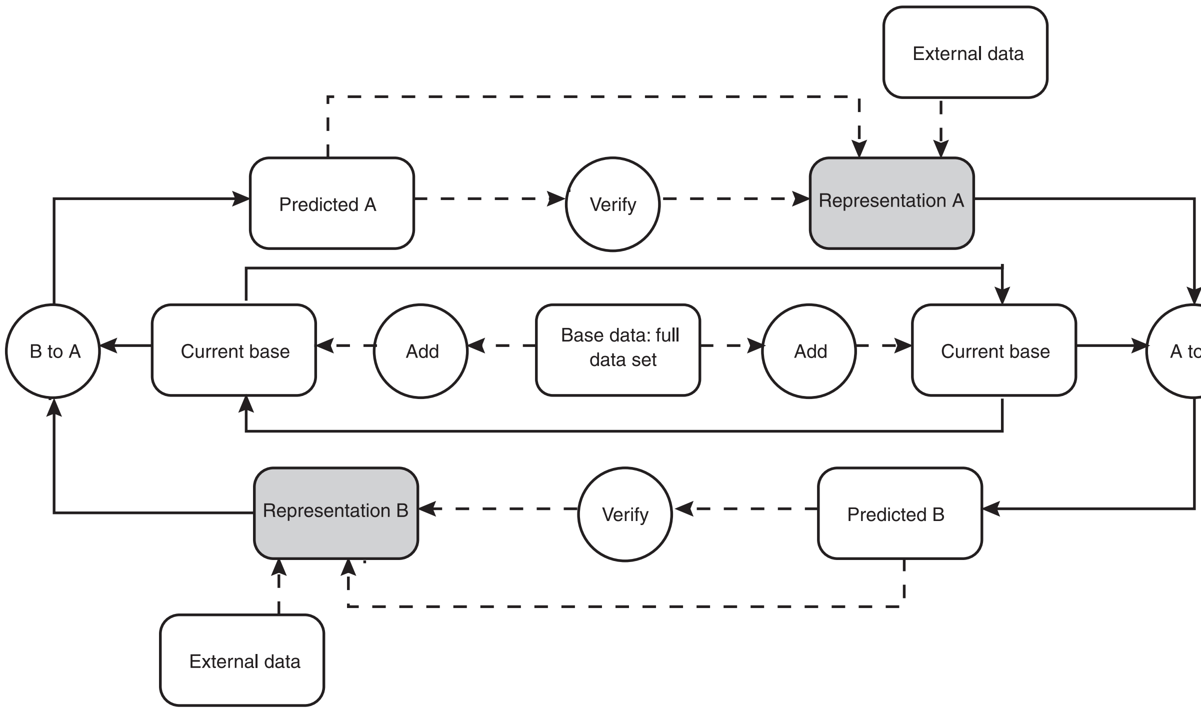

Fig. 1. General bootstrapping concept, using two model representations.

that has two pronunciations: */r iy d/* and */r eh d/*), or that he or she is uncertain about the status of the word-pronunciation pair. If the word is valid but the pronunciation is wrong, the verifier specifies the correct pronunciation by removing, adding or replacing phonemes in the presented pronunciation. A new audio version is generated, for which the verifier can specify a new verdict. At this stage, the learning algorithm updates the word-to-pronunciation model in order to account for the corrected pronunciation. The process is repeated with updated predictions until a pronunciation lexicon of sufficient size is obtained. A related approach was developed by Maskey *et al.*,[5] achieving comparable results.

During prior experimentation, a number of tools were developed in support of the bootstrapping process, including an efficient grapheme-to-phoneme rule extraction algorithm (Default& Refine[6]), automatic error detection tools[4] and a software system that supports the bootstrapping process. While rule extraction is typically computationally efficient for small lexicons, the time required to extract grapheme-to-phoneme rules from a sizeable lexicon slows the process significantly, especially if continuous rule updating is required. In our experience, the bootstrapping system became noticeably cumbersome at approximately 2000 words. All the lexicons developed during prior experimentation were therefore fairly small (approximately 1000 to 2000 words). To overcome this barrier, a new incremental version of the grapheme rule extraction algorithm was defined that performs local refinement of grapheme-to-phoneme rules, accepting a small degradation in learning efficiency in exchange for fast update times.[4] Incremental updating is performed continuously, after every word that is edited by the dictionary developer. After a pre-defined interval the system performs a full rule update (a global optimization operation we term 'synchronization'), which is slower but results in more accurate models. The length of the update interval is defined by the dictionary developer and indicates the amount of 'model drift' that is allowed. (The length of a typical update interval is 50 to 100 corrected words; note that the number of actual words added to the lexicon during this interval is larger.) In this paper, we evaluate these tools during the development of a medium-sized pronunciation lexicon in a resource-scarce language, during a continuous process that starts well beyond the previous 2000-word barrier.

Bootstrapping framework

During bootstrapping, a model is grown systematically, becoming increasingly accurate from one increment to the next. When analysing the bootstrapping process, it soon becomes apparent that the process relies on an automated mechanism to convert among various representations of the model considered. Each representation describes the same task in a format that provides a specific benefit, either because the representation is amenable to automated modelling and analysis or because it describes the current model in a way that is convenient for a human to verify and improve. For example, during the bootstrapping of pronunciation models, the correctness of an entry in the pronunciation lexicon (the first representation) is easily verified by a human, whereas automated analysis of the extracted grapheme-to-phoneme rules (the second representation) can be used to identify possible errors that require re-verification. In contrast, during the bootstrapping of acoustic models for speech recognition, both representations are amenable to automated analysis and updating: the acoustic models (the first representation) are typically built from the phonemic segmentation of the audio data (the second representation) through automated training, clustering and re-training of acoustic models, while the phonemic segmentation of the audio data are created through automated Viterbi alignment of the phonemic transcriptions, utilizing the current acoustic models. By converting back and forth between these two representations in a bootstrapping fashion, both the phonemic segmentations and the acoustic models become increasingly accurate.

Bootstrapping components

The general bootstrapping concept using two model representations is depicted in Fig. 1. The number of representations is limited to two for the sake of simplicity — three or more representations can also be included in the model.

The following components play a role during bootstrapping:

- *Alternative representations*: two or more representations of the same model lie at the heart of the bootstrapping process. In Fig. 1 these are indicated as A and B. During the bootstrapping of pronunciation models, one of these representations (say, 'A') can easily be verified by a human, while the second representation (say, 'B') is more amenable to verification through automated analysis. For other applications, either none or both representations may be amenable to either type of verification.
- *Conversion mechanisms*: each conversion mechanism (indicated as A→B and B→A) provides an automated or semi-automated means to convert data from one representation to another.
- *Verification mechanisms*: once converted to a specific representation, the model can be improved via automated or human (manual) verification, indicated in the figure by the 'Verify' components.
- *Base data*: this term is used to refer to the domain of the model. The current base indicates the domain that has been used in training the current model, and consists of a subset of or the full base data set. The current base data are implicitly or explicitly included in each of the two representations.
- *Increment mechanisms*: the Add components are used to increase the current base during bootstrapping. At the one extreme, all model instances can be included in a single increment; at the other, a single instance can be added per bootstrapping cycle. As a possible extension, the increment mechanisms may use active learning techniques[7,8] in order to select an appropriate set of instances to add.
- *External data*: this term refers to additional data sources that are used during bootstrapping. Typically, external data are used to initialize a bootstrapping system with models that were developed on a related task.

Bootstrapping process

Prior to bootstrapping, the various representations are initialized in preparation for the first iteration. Typically only a single representation requires initialization (A in this instance). External data may be included in this process, or the bootstrapping process starts without any initial knowledge of the task not included in the base data. The increment mechanism chooses the first base set to use. Once initialized, the bootstrapping process consists of the following steps, many of which are optional in the general case, as indicated:

1. The current base, as well as the current representation A, is used to generate the next representation B. During pronunciation modelling, A consists of word-pronunciation pairs and B of grapheme-to-phoneme rules.
2. B is optionally verified, either manually or automatically. During pronunciation modelling, this step is typically omitted, unless the system is requested to flag possible errors for re-verification.
3. Based on the current state of the bootstrapping system, the increment mechanism increases the current base set. During pronunciation modelling, additional words are added to the word list being considered.
4. The current base as well as the current representation B is used to generate representation A.
5. A is optionally verified, either manually or automatically. During pronunciation modelling, this step is not optional and consists of human verification of the newly generated word-pronunciation pairs.
6. Based on the current state of the bootstrapping system, the increment mechanism increases the current base set. (In the general case either step (3) or (6) is optional. During pronunciation modelling, step (6) is omitted.)

This cycle is repeated until a sufficiently accurate and/or comprehensive model is obtained. Note that while step (5) is required during the bootstrapping of pronunciation models, both steps (2) and (5) may be optional for other applications (such as the bootstrapping of acoustic models described above) where model improvement occurs during the automated representation conversion process rather than through post-conversion verification.

Bootstrapping efficiency

The main aim of a bootstrapping system is to obtain as accurate a model as possible from available data. When human intervention is used to supplement or create the training data themselves, the aim shifts towards *minimizing the amount of human effort required* during the process. This is the focus of our analysis, and we therefore measure bootstrapping efficiency as a function of model accuracy:

$$Efficiency(a) = \frac{t_{bootstrap}(a)}{t_{manual}(a)}, \tag{1}$$

where a is the accuracy of the current model as measured against an independent test set and $t_{bootstrap}(a)$ and $t_{manual}(a)$ specify the time (measured according to amount of human intervention) required to develop a model of accuracy a with and without bootstrapping, respectively.

Bootstrapping is analysed according to bootstrapping cycles. While bootstrapping, all base instances do not result in valid data that can be included in the model training process. Of the instances that define a valid base data, some will be correctly represented by the initial representation (B), and others will contain errors. Recall from the Background section that the verifier can specify four verdicts per word: identifying the word-pronunciation pair as *invalid, ambiguous, uncertain* or *correct,* possibly after correcting the pronunciation manually. We define a number of variables to assist us in the analysis of these instances: At the start of cycle x of the bootstrapping process, we define $n(x)$ as the number of instances included in the current base, $n_{invalid}(x)$ as the number of instances that are invalid, ambiguous or uncertain; $n_{correct}(x)$ as the number of instances that are valid and correct, and $n_{error}(x)$ as the number of instances that are valid and incorrect. For these variables, the following will always hold:

$$\begin{aligned} &n(x) = n_{invalid}(x) + n_{valid}(x), \\ &n_{valid}(x) = n_{correct}(x) + n_{error}(x). \end{aligned} \tag{2}$$

Related incremental variables are used to represent the increase of word-pronunciation pairs of a specific type during cycle x, namely $inc\_n(x)$, $inc\_n_{invalid}(x)$, $inc\_n_{valid}(x)$, $inc\_n_{correct}(x)$ and $inc\_n_{error}(x)$. The same intervention mechanism may have different cost implications based on the status of the instance. In the simplest case, the status of an instance may simply be correct, incorrect or invalid, but subtler differences are possible, e.g. the number of changes required to move from an incorrect to a correct version. The expected status of a newly predicted instance changes as the system becomes more accurate. Prior to human intervention at stage x of the bootstrapping process, the number of instances of each status within the current increment is given by:

$$\begin{aligned} &inc\_n(x) = \sum_{s \in status} inc\_n(s,x), \\ &status \in \{invalid, correct, error_1, error_2, error_3 \ldots\}. \end{aligned} \tag{3}$$

where $error_x$ indicates that the current pronunciation (prior to verification) contains x phoneme errors, that is, x substitutions, deletions or insertions are required to correct the pronunciation. Note that the correct status is similar to the *error*$_0$ status.

Combining machine learning and human intervention in a way that minimizes the amount of human effort required during the process can be achieved in two ways: (a) by minimizing the effort required by the human verifier to identify errors accurately, and (b) by optimizing the speed and accuracy with which the system learns from the human input. The remainder of this section describes the various factors that influence the efficiency of the bootstrapping process from both these perspectives. By analysing these factors individually, it becomes possible to understand better the effect on the overall process when optimizing any single factor, and it also becomes possible to predict better the effort involved when bootstrapping a specific resource of a specific size.

Human factors

The first human factor that impacts on the efficiency of the bootstrapping process relates to required user expertise: whether the task requires expert skills, or whether a limited amount of task-directed training is sufficient. If it is assumed that the user has the skills required, the following measurements provide an indication of the efficiency of the bootstrapping process for a specific user:

- *User learning curve*: the time it takes for a specific user to become fully proficient using the bootstrapping system. Measured as t_{train}, initial training data are assumed to be discarded.
- *Cost of intervention*: the average amount of user time required per intervention *i* when an instance has status *s*, for a fully trained user using the bootstrapping system. Measured as $t_{verify}(i, s)$, a different average cost may be associated with different types of interventions. If more than one intervention is used to generate a single instance during one cycle of bootstrapping, the combination of mechanisms is modelled as an additional (single) mechanism. Depending on the bootstrapping process, it may be more realistic to measure this value for a set of instances. During pronunciation modelling, there are only two types of interventions: verifying predictions and verifying the list of errors.
- *Task difficulty*: the average number of errors generated by a fully trained user using the bootstrapping system. Indicated by $error\_rate_{bootstrap}(i, s)$, this is measured as the percentage of errors generated by a user using intervention mechanism *i* to verify an instance initially in state *s*. For example, when a single predicted pronunciation that contains 2 errors is verified, the $error\_rate_{bootstrap}(verify_{single}, error_2)$ is less than 0.5%.
- *Quality and cost of user verification mechanisms*: implicit in the above two measurements are the cost and effect on error-rate of additional assistance provided during user intervention. Rather than modelling additional user assistance provided during existing interventions separately, the combined intervention is again modelled as an additional type of intervention. In the same way, automated verification mechanisms are modelled as additional interventions.
- *Difficulty of manual task*: The average number of errors for a fully trained user developing instances manually. Indicated by $error\_rate_{manual}$, this is measured in percentage as the average number of errors per 100 manual instances developed, where each manual instance can be associated with an individual base data instance in the bootstrapped system.
- *Manual development speed*: The average amount of time per instance development for a fully trained user performing this task manually, measured as $t_{develop}$.
- *Initial set-up cost*: The time it takes for a user to prepare the initial system for manual development or bootstrapping, measured in time as t_{setup_manual} and $t_{setup\_bootstrap}$, respectively.

Machine learning factors

The faster a system learns between verification cycles, the fewer corrections are required from a human verifier, and the more efficient the bootstrapping process becomes. From a machine learning perspective, learning speed and accuracy are directly influenced by the following factors:

- *Predictive accuracy of current base*: modelled as the expected number of instances of each status at a specific cycle of the bootstrapping process, and indicated by $E(inc\_n(s, x))$. Implicit in this measurement are four factors:—
  - —*Accuracy of representations*: the ability of the chosen representations to model the specific task.
  - —*Set sampling ability*: the ability to identify the next 'best' instance or instances to add to the knowledge base, possibly using active learning techniques.
  - —*System continuity*: the speed at which the system updates its knowledge base. This has a significant effect on system responsiveness, especially during the initial stages of bootstrapping.
  - —*Robustness to human error*: the stability of the conversion mechanisms and chosen representations in the presence of noise introduced by human error.
- *On-line conversion speed*: any additional time costs introduced when computation is performed while a human verifier is required to be present (but idle while waiting for the computation to complete). It is measured as the average time per number of valid instances developed and indicated by $t_{idle}(n)$.
- *Quality and cost of verification mechanisms*: the average amount of time required to utilize additional assistance mechanism *j* — from a computational perspective — when an instance is in status *s*, measured as $t_{auto}(j, s)$.
- *Validity of base data*: Using invalid data slows the bootstrapping process, especially if human intervention is required to verify the validity of base data. This is measured in percentage of base data, and is indicated by *valid_ratio.*

Two additional factors that are not included explicitly in the general model, but can be included based on the requirements of the specific bootstrapping task, are:

- *Conversion accuracy*: the ability of the conversion model to convert between representations without loss of accuracy.
- *Effect of incorporating additional data sources*: the ability of the system to boost accuracy by incorporating external data sources at appropriate times.

System analysis

The combined effect of the machine learning factors and human factors provides an indication of the expected cost of using the bootstrapping system. The time to develop a bootstrapping model via *N* cycles of bootstrapping, using a set of interventions *I*, is given by:

$$\begin{aligned} t_{bootstrap}(N,I) &= t_{setup\_bootstrap} + t_{train} + t_{iterate}(N,I) \\ &= t_{setup\_bootstrap} + t_{train} \\ &\quad + \sum_{x=1}^{N-1}\left(\sum_{i \in I} \sum_{s \in status} (t_{verify}(s,i) + t_{auto}(s,i)) * inc\_n(s,x) \right. \\ &\quad \left. + t_{idle} * inc\_n_{valid}(x+1) \right) \end{aligned} \qquad (4)$$

where $t_{iterate}(N, I)$ combines the cost of the various iterations, excluding the cost associated with system setup and user training. The expected value of *inc_n*(s, x) depends on the specific conversion mechanism, and is influenced by *valid_ratio* and $error\_rate_{bootstrap}(i, s)$.

This cost of bootstrapping can be compared with the expected cost of developing n_{manual} instances via a manual process:

$$t_{manual} = t_{setup\_manual} + t_{develop} * n_{manual}. \quad (5)$$

If $n_{bootstrap}$ and n_{manual} are chosen such that

$$E[inc\_n(correct, n_{bootstrap})] = E[inc\_n(correct, n_{manual})], \quad (6)$$

where the number of valid instances generated during bootstrapping is given by:

$$n_{bootstrap} = \sum_{x=1}^{N-1} inc\_n(valid, x), \quad (7)$$

the accuracy of each of the two systems is approximately equivalent, and the values of Equations (4) and (5) can be combined according to Equation (1) in order to obtain a measure of the expected efficiency of the bootstrapping process.

Bootstrapping a medium-sized lexicon

Experimental approach

We use the above framework to improve and analyse the efficiency of our pronunciation model bootstrapping process. During prior experimentation, a number of bootstrapped lexicons were created, each less than 2000 words. In this work we first cross-analyse these prior lexicons and manually verify discrepancies in order to obtain a verified lexicon of approximately 5000 words. We then perform bootstrapping according to the bootstrapping framework described above, and initialize the bootstrapping system using the verified lexicon. We use a linguistically sophisticated first language speaker (Developer 'C') who has previous experience in using the bootstrapping system to develop the lexicon and measure the time taken for each action[b]. We obtain a word list from random text from the Internet, and order words randomly (in the list of new words to be predicted). We use incremental Default&Refine for active learning in between synchronization sessions and standard Default&Refine during synchronization. We set the update interval (number of words modified between synchronizations) to 50.

At the end of the bootstrapping session we perform error detection. (No additional error detection is performed during bootstrapping.) We first extract the list of graphemic nulls, and identify possible word errors from the graphemic null generators.[c] We then extract Default&Refine rules from the full lexicon with the purpose of using these rules to identify errors, similar to the process described in our earlier work.[4] Only words that generate new grapheme-to-phoneme rules are considered for manual re-verification. We list as possible errors all words from word sets that result in a new rule and contain fewer than five words, and verify these words manually[d].

Results

The parameters of the bootstrapped lexicon are listed in Table 1. We measure the time taken by the verifier to perform each verification action, and analyse the effectiveness of the verification process from a human factors perspective. Figure 2 illustrates the verification process as the lexicon grows from 5500 to 7000 words. We plot the time taken to verify each valid word, indicating whether 0, 1, 2 or 3 corrections are required for each word as it is added to the lexicon. (The number of training words on the x-axis includes both valid and invalid words.)

Table 1. Parameters of the Afrikaans bootstrapped lexicon.

Number of graphemes in orthography	40
Number of phonemes in phoneme set	43
Number of words in starting lexicon	4923
Total number of words in bootstrapped lexicon	8053
Number of valid words in bootstrapped lexicon	7782
Number of derived rules (*Default&Refine*)	1471

We note the following:

- *User learning curve*: the verifier was proficient in using the system prior to the current bootstrapping session, and further training was not required. The value of t_{train} during earlier sessions was <120 min.
- *Cost of intervention*: in this experiment we used two intervention mechanisms: verifying predictions and verifying the list of possible errors. Table 2 provides the average verification times observed for Developer C where the intervention mechanism is a single verification of a prediction ($t_{verify}(single, s)$) for words that are in different states s prior to verification. Verification of the list of possible errors took approximately 27 minutes (for approximately 3000 words).
- *Task difficulty*: during the bootstrapping process, 3019 words were added to the lexicon, of which 181 were invalid or ambiguous. During error detection, 9 errors were found in the remaining 2838 valid words. Given our previous analysis,[4] we estimate that this represents at least 50% of the errors, and therefore estimate the actual error rate to be 0.6%[e]. It is interesting to note that, while our error detection protocol resulted in a re-verification of 3.3% of the full lexicon (1832 grapheme-specific patterns, or about 300 words), the average position of each error in the ordered error prediction list was at 0.67% of the full training lexicon, with the majority of errors found in the first 0.1% of words, that is, the first or second pattern on the per-grapheme list of potential errors.

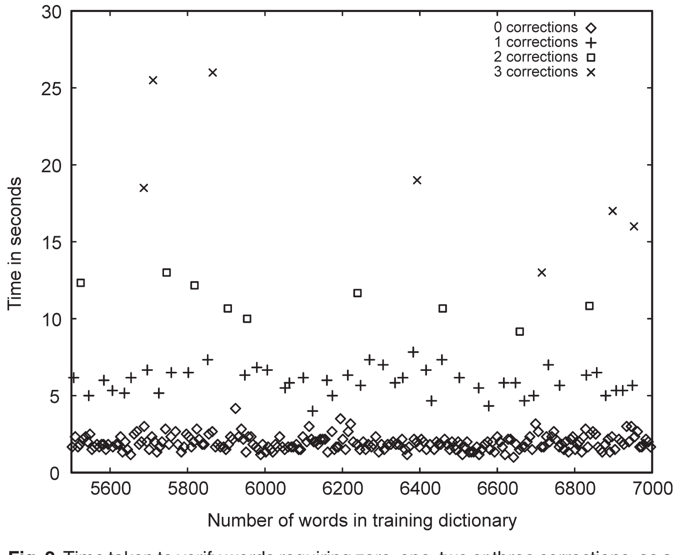

Fig. 2. Time taken to verify words requiring zero, one, two or three corrections, as a function of the number of words verified. For the first three measures, the averages were computed for blocks of five words each.

[b]In prior work we have shown that even linguistically unsophisticated user can be trained to use the bootstrapping system with high accuracy.[6]

[c]Graphemic nulls are inserted during alignment where a single grapheme maps to more than one phoneme. See ref. 16 for a detailed description of graphemic nulls and graphemic null generators.

[d]A word set associated with a rule tends to have either only one or two words associated with it, or a large set of words: within an acceptable range, the error detection process is not sensitive with regard to the exact cut-off point selected.

[e]18 errors in 2838 valid words.

Table 2. Statistics of the time taken to verify words requiring 0, 1, 2 or 3 errors, or to identify a word as invalid or ambiguous (μ is the mean, and σ the standard deviation.).

Verdict	Time in seconds	
	μ	σ
Correct	1.95	1.35
1 error	5.79	2.3
2 errors	10.74	3.19
3 errors	17.91	6.12
Invalid	3.39	4.71
Ambiguous	8.92	5.08

- *Difficulty of manual task*: $error\_rate_{manual}$ is assumed to be <0.5%, which is an optimistic estimate for the range of manual development speeds evaluated.
- *Manual development speed*: different values of $t_{develop}$ are used for comparison, ranging from 19.2 s, again an optimistic estimate.
- *Initial set-up cost*: As this is an extension of an existing system, no further set-up cost was incurred[f].

From a machine learning perspective, the following is observed:

Predictive accuracy of current base: measured directly during experimentation, the number of corrections required per word added to the lexicon ($inc\_n(s, n)$) is depicted in Fig. 3. We plot the running average (per blocks of 50 words) of the number of corrections as a function of the number of words verified.

On-line conversion speed: the average time taken for a synchronization event was 50.15 seconds (σ = 7.72 s). This value increased gradually from 35 s during the initial cycle, to 56 s in the final cycle.

Quality and cost of verification mechanisms: the computational times required for both verification mechanisms are included in the verification times. No additional processing is required.

Validity of base data: 94% of the base data was valid.

Based on our observations during this experiment, we can assign approximate values to the different costs and efficiencies involved during bootstrapping of an Afrikaans lexicon up to 10 000 words. We list these values in Table 3.

Using the above observations, we obtain the following expected cost of N cycles of bootstrapping:

$$E[t_{bootstrap}(N)] = E[t_{setup\_bootstrap}] + E[t_{train}] + E[t_{iterate}(N)], \tag{8}$$

where:

$$\begin{aligned} E[t_{iterate}(N)] = & \sum_{x=1}^{N-1}\left(\sum_{s\in\ status}(E(t_{verify}(s, single))*E(inc\_n(s,x))\right) \\ & + \sum_{x=1}^{N-1}\Big(t_{idle}(inc\_n(valid, x+1)) + \\ & t_{verify(error-det)}(inc\_n(valid, x+1))\Big). \end{aligned} \tag{9}$$

We assume an update event after every 100 errors (approximately 400 words verified.) As t_{idle} is dominated by $t_{verify}(error\text{-}det)$ during the initial 10 000 words, we keep this value constant as the number of words in the training lexicon increases,[g] and estimate it at:

$$t_{verify(error-\det)}(400) + t_{idle}(400) = 180 \text{ seconds}. \tag{10}$$

From Table 3 we estimate $E(t_{verify}(s, single))$ as $t_0 + t_e x$ seconds, where x is an indication of the number of corrections required, t_0 = 2 and t_e = 4.5.

[f]In the previous experiment $t_{setup\_bootstrap} - t_{setup\_manual}$ <60 min.

[g]This value is influenced by the number of words corrected per cycle – a number that remains constant per cycle.

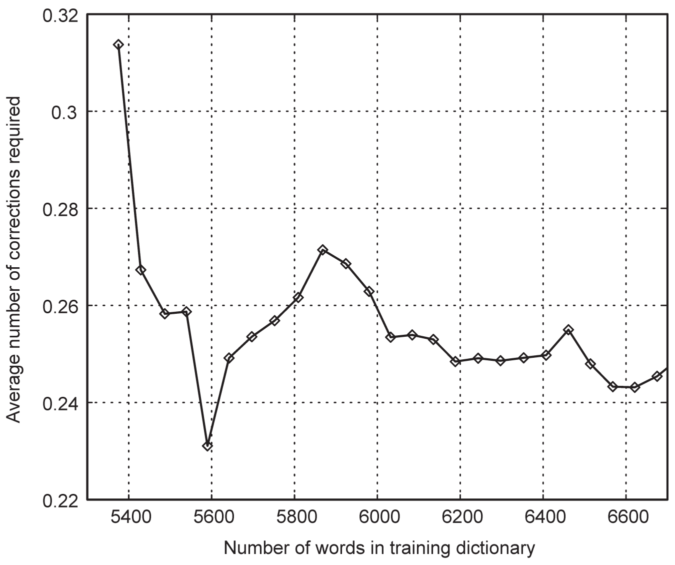

Fig. 3. The average number of corrections required per word as a function of the number of words verified. Averages were computed for blocks of 50 words each.

In order to estimate $\sum_{x=1}^{N-1} E(t_{verify}(s, single))E(inc\_n(s,x))$ for different states (different numbers of corrections per word), we smooth the number of errors across the training data — as if a word could have only one error — and fit an exponential curve through the accuracy measurements depicted in Fig. 3. That is, we assume the probability that the system will predict an error when the training lexicon is of size d is given by $p_e(d)$, where:

$$\begin{aligned} & p_e(d) = P_0 e^{-\frac{d}{k}}, \\ & \text{i.e. } \log p_e(d) = \log P_0 - \frac{d}{k}, \end{aligned} \tag{11}$$

and P_0 and k are parameters to be estimated. The time required for d corrections $T(d)$(excluding synchronization events) is then given by:

$$\begin{aligned} T(d) &= \sum_{i=0}^{d-1}(t_0 + t_e P_0) \\ &= dt_0 + t_e P_0 \sum_{i=0}^{d-1} e^{-\frac{d}{k}} \\ &= dt_0 + t_e P_0 \frac{1 - e^{-\frac{d}{k}}}{1 - e^{-\frac{1}{k}}}. \end{aligned} \tag{12}$$

For the specific data depicted in Fig. 3 we obtain the estimates:

$$\begin{aligned} & \log P_0 = -1.274 \\ & -\frac{1}{k} = -3.49 * 10^{-5}. \end{aligned} \tag{13}$$

We can combine Equations (9) and (12) in order to estimate the value of $E[t_{iterate}(d/400)]$ for various values of total lexicon size d:

$$\begin{aligned} E[t_{iterate}(d/400)] = & dt_0 + t_e P_0 \frac{1 - e^{-\frac{d}{k}}}{1 - e^{-\frac{1}{d}}} \\ & + \frac{d}{400} * (t_{verify(error-det)}(400) + t_{idle}(400)). \end{aligned} \tag{14}$$

In Fig. 4 we plot Equation (14) for different values of d, using the estimates from Equations (10) and (13). On the same graph we plot the cost of manual lexicon development (again excluding set-up cost) using estimates for $t_{develop}(d)$ of 19.2 and 30 s, both optimistic estimates. For these estimates we assume that the same base data (or at least data with a similar validity ratio) are

Table 3. Observed values for various bootstrapping parameters.

Bootstrapping parameter		Estimated value
Training cost	t_{train}	<120 min
Verification cost for single words, with *x* corrections required for a word in state *s*:	$t_{verify}(single, s)$	(2 + 4.5x) s
Verification cost during error detection (per 1000 words):	$t_{verify}(error\text{-}det)$	<10 min
Verification cost during error detection (per 400 words):	$t_{verify}(error\text{-}det)$	<3 min
Task difficulty–bootstrapping, no error detection	$error\_rate_{bootstrap}$	0–1%
Task difficulty–bootstrapping, error detection	$error\_rate_{bootstrap}$	0–0.5%
Task difficulty–manual	$error\_rate_{manual}$	0-0.5%
Manual development speed	$t_{develop}$	19.2–30 s
Initial set-up cost	$t_{setup\_bootstrap} - t_{setup\_manual}$	<60 min
Update interval		100 errors (~400 words)

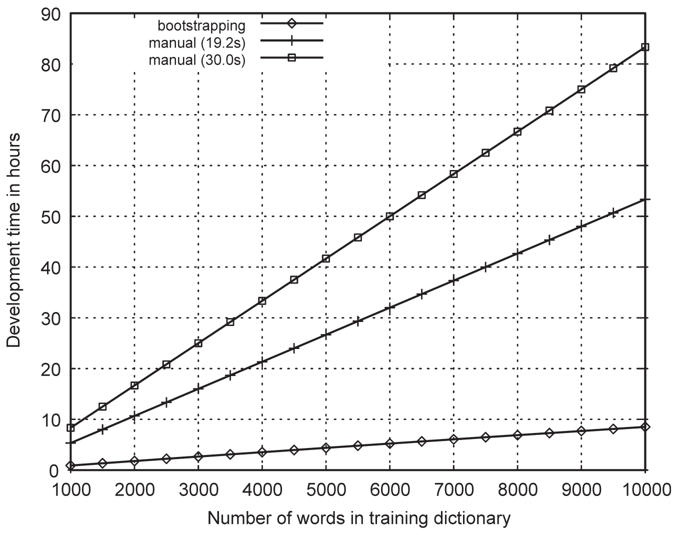

Fig. 4. Time estimates for creating different sized lexicons. Manual development is illustrated for values of $t_{develop}(1)$ of 19.2 and 30 seconds, respectively.

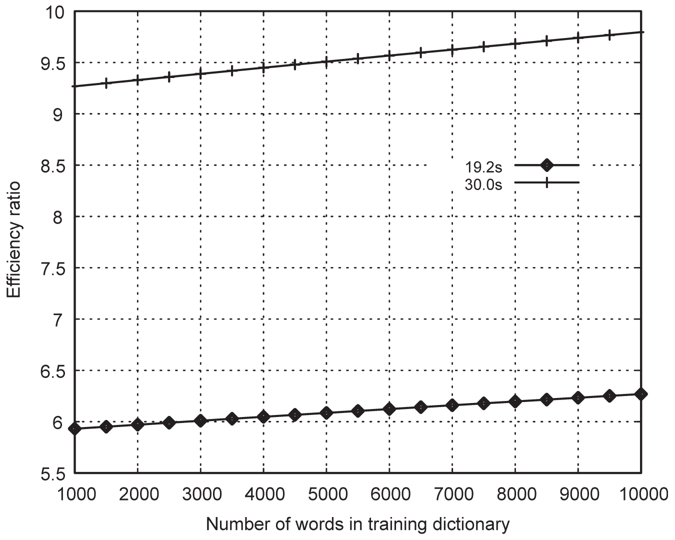

Fig. 5. Estimates of the efficiency of bootstrapping, as compared with manual development for values of $t_{develop}(1)$ of 19.2 and 30 seconds, respectively.

used for both approaches. We also assume that the error rates for the bootstrapping system with error detection and the manual process are approximately equal. As can be seen from the figure, the bootstrapping approach requires significantly less effort than the manual approach. A manual developer would have to be able to generate a correct instance every 3 seconds, in order to develop a comparable lexicon within the same time frame. In Fig. 5 we plot the efficiency estimates of the bootstrapping process as compared to a manual lexicon development process for the same values as Fig. 4.

Conclusion

A pronunciation model is one of the key components required for the development of speech technology in a resource-scarce language. In this paper we developed a framework for the analysis of a typical bootstrapping process, and demonstrated the practical application of such a framework during the bootstrapping of a pronunciation model in a resource-scarce language. We reported on results for a lexicon that is large enough to be of practical use, with final results summarized in Figs 4 and 5.

Upon completion of the Afrikaans pronunciation lexicon, two further bootstrapped pronunciation lexicons were developed in Zulu and Sepedi, respectively. The Zulu lexicon has since been integrated in a general-purpose text-to-speech (TTS) system developed in the Festival[9] framework. This was accomplished as part of the Local Language Speech Technology Initiative (LLSTI),[10] an internationally collaborative project that aims to support the development of speech technology systems in local languages. A small grapheme-to-phoneme rule set (84 rules from 855 words) was generated using the bootstrapping system and converted to the Festival letter-to-sound format. The TTS system used the Multisyn approach to synthesis and is described in more detail elsewhere.[11,12] The completed system was evaluated for intelligibility and naturalness by both technologically sophisticated and technologically unsophisticated users.[13]

The Sepedi lexicon (Sepedi is a dialect of Northern Sotho) was developed and integrated in an automatic speech recognition (ASR) system in collaboration with partners from the University of Limpopo. Again a fairly small number of words were bootstrapped in order to develop a concise set of letter-to-sound rules (90 rules from 2827 words). These were then used to develop[14] a speech recognition system using the HTK[15] framework.

The bootstrapping approach has been shown to be highly efficient for the development of pronunciation lexicons. We foresee that further application of the approach developed will ensure the availability of pronunciation models suitable for speech processing systems in all of South Africa's official languages.

1. Barnard E., Cloete J.P.L. and Patel H. (2003). Language and technology literacy barriers to accessing government services. *Lecture Notes in Computer Science* **2739**, 37–42.
2. Davel M. and Barnard E. (2003). Bootstrapping for language resource generation. In *Proc. Annual Symposium of the Pattern Recognition Association of South Africa*, pp. 97–100.
3. Davel M. and Barnard E. (2004). The efficient creation of pronunciation dictionaries: human factors in bootstrapping. In *Proc. Interspeech*, pp. 2797–2800. Jeju, Korea.
4. Davel M. and Barnard E. (2005). Bootstrapping pronunciation dictionaries:

practical issues. In *Proc. Interspeech,* Lisbon, Portugal.
5. Maskey S., Tomokiyo L. and Black A. (2004). Bootstrapping phonetic lexicons for new languages. In *Proc. Interspeech,* pp. 69–72. Jeju, Korea.
6. Davel M. and Barnard E. (2004). A default-and-refinement approach to pronunciation prediction. In *Proc. Annual Symposium of the Pattern Recognition Association of South Africa,* pp. 119–123.
7. Cohen D., Ghahramani Z. and Jordan M. (1990). Active learning with statistical models. *J. Artific. Intell. Res.* **4**, 129–145.
8. Seeger M. (2002). Learning with labeled and unlabeled data. Technical Report, Institute for Adaptive and Neural Computation, University of Edinburgh.
9. Black A., Taylor P. and Caley R. (1999). The Festival Speech Synthesis System. Online: http://festvox.org/festival/
10. Local Language Speech Technology Initiative (LLSTI) (2005). Online: http://www.llsti.org
11. Davel M. and Barnard E. (2004). LLSTI isiZulu TTS Project Report. CSIR, Pretoria.
12. Louw J.A., Davel M. and Barnard E. (in press). A general purpose isiZulu TTS system. *South African Journal of African Languages.*
13. Davel M. and Barnard E. (2004). LLSTI isiZulu TTS Evaluation Report. CSIR, Pretoria.
14. Modiba T.M. (2004). *Aspects of automatic speech recognition with respect to Northern Sotho.* M.Eng. thesis, University of the North, South Africa.
15. Young S., Kershaw D., Odell J., Ollason D., Valtchev V. and Woodland P. (2000). *The HTK Book. Revised HTK Version 3.0.* Online: http://htk.eng.cam.ac.uk/
16, Davel M. (2005). *Pronunciation modelling and bootstrapping.* Ph.D. thesis, University of Pretoria, South Africa.