A TURBO-STYLE ALGORITHM FOR LEXICAL BASEFORMS ESTIMATION

Ghinwa F. Choueiter, Mesrob I. Ohannessian, Stephanie Seneff, and James R. Glass

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA 02139
{ghinwa, mesrob, seneff, glass}@mit.edu

ABSTRACT

In this research, an iterative and unsupervised Turbo-style algorithm is presented and implemented for the task of automatic lexical acquisition. The algorithm makes use of spoken examples of both spellings and words and fuses information from letter and subword recognizers to boost the overall lexical learning performance. The algorithm is tested on a challenging lexicon of restaurant and street names and evaluated in terms of spelling accuracy and letter error rate. Absolute improvements of 7.2% and 3% (15.5% relative improvement) are obtained in the spelling accuracy and the letter error rate respectively following only 2 iterations of the algorithm.

Index Terms**—** Turbo-style, spelling, pronunciation, lexical acquisition

1. INTRODUCTION

In speech recognition systems, automatic lexical update is the process of introducing new entries into the phonetic dictionary as well as refining pre-existing ones. Such an update process can be triggered by newly acquired information such as a spoken example of an unknown word or its spelling. The capability of automatically learning a reliable estimate of a lexical entry (both spelling and phonetic baseform) of a word from spoken examples, can prove quite beneficial. For example, consider spoken dialogue systems, which have been slowly emerging as a natural solution for information retrieval applications [1]. Such systems often suffer from dialogue breakdown at critical points that convey important information such as named entities or geographical locations. One successful approach proposed for error recovery in dialogue systems lies in speak-and-spell models, that prompt the user for the spelling of an unrecognized word [2, 3]. In such cases, both the spoken spelling and word are available. The question that this research attempts to answer is: Given both the spoken spelling and spoken word how well can a valid lexical entry in a dictionary be learnt?

This research introduces an unsupervised iterative technique denoted *Turbo-style algorithm* and applies it to the task of automatic lexical acquisition. In particular, spoken examples from two complementary domains, spelling and pronunciation, are presented to a letter and subword recognizer respectively. The output of each recognizer is then processed by a bi-directional letter-to-sound (L2S) model and injected back into the other recognizer in the form of *soft* bias information. Such a set-up is denoted Turbo-style learning algorithm since it is inspired by the principles of Turbo Codes [4]. The term Turbo Code is in turn a reference to turbo-charged engines where part of the output power is fed back to the engine to improve the performance of the whole system.

There has been significant research on automatic lexical generation [5, 6, 7]. However, the novel contribution of this work is two-fold: (1) Spoken examples of both the spelling and the word are used as opposed to the word only, and (2) a bi-directional L2S model is used to exchange bias information between the spelling and pronunciation domain so as to boost the overall performance of the tandem model. It is worth noting that the set-up does not consult a lexicon when estimating the spelling.

The basic principle of the proposed algorithm is the fusion of several sources of information, and it can be generalized to different set-ups. For example, a recent approach to unsupervised pattern discovery in speech produces reliable clusters of similar speech patterns [8]. The generated clusters can be processed by multiple subword recognizers whose outputs can be fused to boost the pronunciation recognition performance.

In the rest of the paper, the Turbo-style algorithm is described in Section 2, and the implementation components in Section 3. The experimental set-up and parameter tuning are depicted in Sections 4 and 5 respectively. Section 6 reports the results, and Section 7 concludes with a summary.

2. THE TURBO-STYLE ALGORITHM

In this section, the Turbo-style iterative algorithm is presented. The basic principle behind the proposed algorithm is to have two complementary recognizers, spelling and pronunciation, exchange bias information such that the performance of both systems is improved. In this particular implementation, the letter recognizer first generates an *N*-best list, which is projected into the complementary subword domain using a bi-

This research was supported by the Industrial Technology Research Institute (ITRI) in Taiwan.

directional L2S model. The projected *N*-best list is used to bias the subword LM, by injecting into it the pronunciations that best match the estimated spelling. A similar procedure is repeated in the subword domain. The algorithm is illustrated in Figure 1, and the steps for a pair of spoken spelling and word are as follows:

(1) The spoken spelling is presented to the letter recognizer, and a letter N_1-best list is generated.
(2) The letter N_1-best list is processed by the L2S model, and a subword M_1-best list is produced.
(3) A bias subword language model (LM) is trained with the subword M_1-best list, and interpolated with the base subword LM by a factor w_1. The interpolated LM becomes the new base subword LM.
(4) A subword recognizer is built with the new interpolated subword LM, the spoken word is presented to the subword recognizer, and a subword M_2-best list is generated.
(5) The subword M_2-best list is processed by the S2L model, and a letter N_2-best list is produced.
(6) A bias letter LM is trained with the letter N_2-best list, and the bias letter LM is interpolated with the base letter LM by a factor w_2. The interpolated LM becomes the new base letter LM.
(7) A letter recognizer is built with the new interpolated letter LM.
(8) Go back to Step (1).

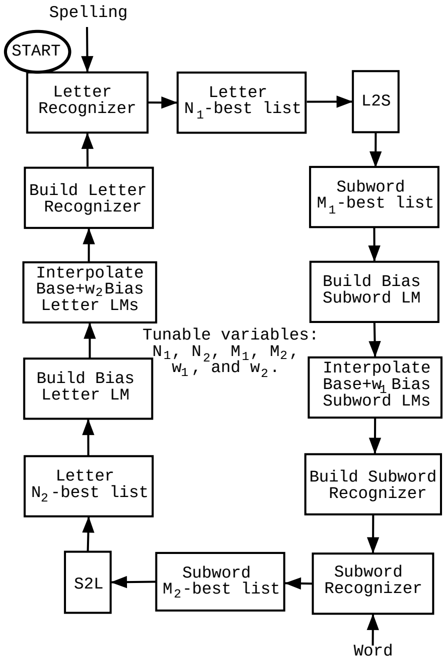

Fig. 1. Illustration of the unsupervised Turbo-style algorithm used to refine the estimates of the spelling and the pronunciation of a new word.

Figure 1 shows 7 parameters that need to be set, N_1, M_1, w_1, N_2, M_2, w_2, and K, the number of iterations performed. The tuning of these parameters is described in Section 5.

3. IMPLEMENTATION COMPONENTS

3.1. The Bi-Directional L2S/S2L Model

The bi-directional L2S model used in this research is based on a context-free grammar (CFG) designed to encode positional and phonological constraints in sub-syllabic structures. The CFG-based subword model is described in detail in [9], and evaluated successfully on the task of automatic pronunciation generation in [10]. Briefly, the CFG describes all possible ways sub-syllabic structures map to subword units as well as all possible ways subword units map to spellings. The CFG pre-terminals are the subword units, which encode only pronunciation information, and the terminals are letter clusters which encode spelling. The total number of pre-terminals and terminals are 677 and 1573 respectively. A by-product of the CFG is an automatically derived mapping between subwords and their spellings, which results in hybrid units, denoted spellnemes[1]. Generating a statistical L2S model is facilitated by the spellneme units. The L2S model, T_{L2U}, is modeled using finite state transducers (FSTs) as follows:

$$T_{L2U} = T_{L2SP} \circ G_{SP} \circ T_{SP2U} \qquad (1)$$

where T_{L2SP} and T_{SP2U} are mappings from letters to spellnemes and from spellnemes to subwords respectively, and G_{SP} is a spellneme trigram. A search through T_{L2U} produces an *N*-best list of pronunciations corresponding to the input spelling. An S2L model is generated similarly.

3.2. The Subword and Letter Recognizers

The subword recognizer is modeled as a weighted FST, R_S:

$$R_S = C \circ P \circ L_S \circ G_S \qquad (2)$$

where C denotes the mapping from context-dependent model labels to context-independent phone labels, P the phonological rules that map phone labels to phoneme sequences, L_S the subword lexicon, which is a mapping from phonemic units to subwords obtained from the CFG, and G_S the subword trigram. A search through R_S produces an *N*-best list of pronunciations corresponding to the spoken word.

The letter recognizer is similarly implemented as a weighted FST, R_L. The letter lexicon, L_L contains 27 entries, the 26 letters of the alphabet and the apostrophe.

4. EXPERIMENTAL SET-UP

The SUMMIT segment-based speech recognition system is used [11] in all the experiments. Context-dependent diphone acoustic models are used with an MFCC (Mel-Frequency Cepstral Coefficient) based feature representation. The diphones

[1] Other researchers have used the term *graphones* for these types of units (e.g. Bisani and Ney [7]).

are modeled with diagonal Gaussian mixture models with a maximum of 75 Gaussians per model, and are trained on telephone speech. The spellneme trigram, G_{SP} used by the L2S model is built with 55k parsed nouns extracted from the LDC pronlex dictionary. The letter trigram, G_L, is trained with 300k Google words, and the subword trigram, G_S, with the same set parsed with the L2S model.

For the purpose of this research, 603 Massachusetts restaurant and street names were recorded together with their spoken spellings. This set is part of a larger data collection effort described in more detail in [10]. The 603 spelling/word pairs are split into a development (Dev) set of 300 pairs and a Test set of 303.

5. PARAMETER TUNING

In this section, the process of setting the parameters of the algorithm is presented. There are various ways of approaching such a problem, and the choice here is to set N_1 and M_2 separately, while M_1 and w_1 are tuned simultaneously, and similarly for N_2 and w_2.

N_1 and M_2 correspond to the number of top candidate spellings and pronunciations generated by the letter and subword recognizers respectively. N_1 is chosen to achieve an effective compromise between capturing the correct spelling and weeding out incorrect ones. This is done by presenting the Dev data to the letter recognizer and monitoring the depth of the correct spelling in the top 100 candidates. By this process, N_1 is empirically set to 20.

In a similar procedure on the pronunciation side, M_2 is empirically set to 50. However, it is worth noting that while reference spellings are available for the letter set-up, no references are available for the subword set-up. To avoid having to manually transcribe subword baseforms, the L2S model is used to automatically generate them [10].

N_2 and w_2 denote the number of top candidate spellings produced by the S2L model and the weight of the letter bias LM respectively. They are tuned to improve the performance of the letter recognizer on the Dev set. Performance is evaluated in terms of *spelling match rate*. A match is when the correct word occurs in the N_1-best list generated by the letter recognizer, where $N_1 = 20$. Since $M_2 = 50$, a subword 50-best list is processed by the S2L, producing a spelling N_2-best list, where N_2 = 20, 100, 500, 1000, 5000, 10000. For each value of N_2, a bias LM is trained with the spelling N_2-best list and interpolated with a base LM. The interpolation weight, w_2 is varied between 0 and 1 in 0.2 steps. For each (N_2,w_2) pair, a letter recognizer is built and the spelling 20-best list is generated. Figure 2 reports the performance as a function of N_2 and w_2, and illustrates that mid-range values of both N_2 and w_2 are best. Based on this, N_2 is set to 1000 and w_2 to 0.4.

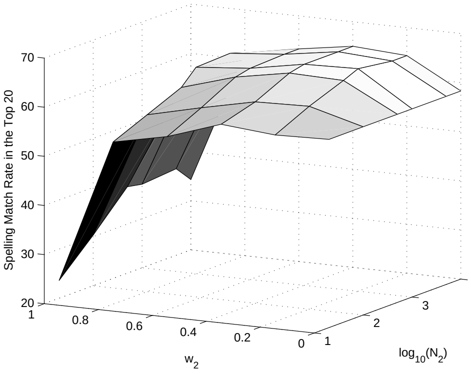

Fig. 2. The spelling accuracy, in a 20-best spelling list, evaluated on the Dev set as a function of N_2 and w_2.

M_1 and w_1 correspond to the number of top candidate subword sequences generated by the S2L model and the weight of the subword bias LM respectively. They are tuned similarly to N_2 and w_2. For lack of space, we only report that M_1 is set to 1000 and w_1 to 0.8. Compared to w_2, the results indicate that the subword recognizer is more confident about the bias information obtained from the letter domain than vice versa. This is expected since the spelling domain is more constrained and hence more reliable than the subword one.

K corresponds to the number of iterations of the Turbo-style algorithm. To set K, the algorithm is run on the Dev set until little change in performance is observed. The results are reported in Table 1 in terms of spelling match rates. The first column is the iteration number, where iteration 0 refers to the initial results prior to receiving any bias information from the complementary domain. The second to fifth columns give the spelling match rates in the top 1, 10, 20, and 100 spelling candidates. The results in Table 1 show substantial improvement in the spelling match rates following iteration 2. For example, the top 1 spelling accuracy improves by an absolute 5.7%. It is noted here that the results of the 0^{th} iteration correspond to the spelling recognizer alone without any feedback from the pronunciation domain. Based on the observation that no significant improvement occurs beyond iteration 3, K is set to 2.

Iteration #	Top 1	Top 10	Top 20	Top 100
0	19.3%	50.6%	57.6%	77.6%
1	24.3%	53.6%	62.3%	78%
2	25%	56.3%	62.6%	76.6%
3	25%	56%	62.6%	76.6%

Table 1. Top 1, 10, 20, and 100 spelling match rates on the Dev set as a function of iterations.

6. PRELIMINARY RESULTS AND DISCUSSION

The parameters are adjusted based on the Dev set as described in Section 5, and preliminary results are obtained on the Test set. Significant improvement is observed in the spelling match rates in Table 2. For example, the top 1 spelling accuracy improves by an absolute 7.2% following 2 iterations. The letter error rate is also found to decrease from 19.3% in iteration 0 to 16.3% in iteration 2 (15.5% relative improvement).

The algorithm also substantially improves the almost-correct spelling rate. In this case, almost-correct spelling is when the edit distance between the top 1 spelling and the correct one is no more than 1 letter. The almost-correct rate increases from 43.2% at iteration 0 to 52.8% at iteration 2. This suggests that a spelling correction has a better chance of finding the reference word in a lexicon retrieved, say from the Web.

No accuracy results are reported in the pronunciation domain due to the lack of a reference. However, Table 3 illustrates dramatic qualitative improvement in the pronunciation of sample words from iteration 0 to iteration 2.

Iteration #	Top 1	Top 10	Top 20	Top 100
0	20.5%	54.1%	66.3%	77.2%
1	26.4%	57.8%	66.9%	80.2%
2	27.7%	59.1%	66.9%	79.2%

Table 2. Top 1, 10, 20, and 100 spelling match rates on the Test set as a function of iterations.

Similarly, Table 4 illustrates sample words and the corresponding spelling improvement from iteration 0 to iteration 2. As shown in Table 4, the bias information obtained from the pronunciation domain could drive the spelling recognizer to a local optimum which does not match the reference, for e.g. *tartufo*, and vice versa. In fact, this phenomenon could explain why very little to no improvement is observed on the Dev set following iteration 3.

Hence, the optimality of the proposed scheme remains to be examined. For example, instead of keeping the parameters N_1, M_2, N_2, w_2, M_1, and w_1 static, it might be more advantageous to adaptively update them to reflect the confidence in the bias information.

7. SUMMARY

In this research, an iterative and unsupervised Turbo-style algorithm has been introduced and implemented for automatic lexical learning. A spoken example of a word and its spelling are presented to a subword and letter recognizer, which recursively exchange bias information through a bi-directional L2S model. As a proof of concept, preliminary experiments were performed using 603 pairs of spoken spellings and words, and results on the 303-pair Test set showed significant absolute improvements of 7.2% and 3% in the spelling accuracy and LER with only 2 iterations of the algorithm.

Word	Iteration 0	Iteration 2
botoloph	-ao+ tf -ow+ l+ -aof	b -owt -axl -aolf
quans	-eyn +z	kw+ -aan +z
olivio	l+ -ey+ df -iy+ -ow+	-axl -iy+ v+ -iy+ -ow+
woodmans	-ahn m+ -aen s+ -ihng	w -uhd m+ -aen +s
churrascaria	jh+ -ehs t -ehr -iy+ -ax+	ch+ -aoer+ -axs k -ehr -iy+ -ax+

Table 3. Sample pronunciations (in subword units) at iterations 0 and 2.

Word	Iteration 0	Iteration 2
mcmenamy	mcnenanys	mcmenamys
tartufo	cruso	cartufo
terranova	trialve	trianove
helmand	heelmand	helmand
scutra	setra	scutra

Table 4. Sample spellings at iterations 0 and 2.

Within the same Turbo framework, it remains important to investigate (1) different schemes for parameter tuning, (2) other methods for exchanging bias information between different domains, as well as (3) extensions of this algorithm to more general set-ups. The algorithm is also expected to be incorporated into a spoken dialogue system for automatically acquiring new words.

8. REFERENCES

[1] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington, “Jupiter: A tephone-based conversational interface for weather information,” *IEEE Trans. on Speech and Audio Proc.*, vol. 8, pp. 85–96, 2000.

[2] H. Schramm, B. Rueber, and A. Kellner, “Strategies for name recognition in automatic directory assistance systems,” *Speech Communication*, vol. 31, pp. 329–338, 2000.

[3] E. Filisko and S. Seneff, “Developing city name acquisition strategies in spoken dialogue systems via user simulation,” in *Proc. SIGDIAL*, Lisbon, Portugal, 2005, pp. 144–155.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error-correcting coding and decoding: Turbo-codes,” in *Proc. ICC*, Geneva, Switzerland, 1993, pp. 1064–1070.

[5] L. R. Bahl et. al., “Automatic phonetic baseform determination,” in *Proc. ICASSP*, Toronto, Canada, 1991, pp. 173–176.

[6] L. Galescu and J. Allen, “Name pronunciation with a joint n-gram model for bi-directional grapheme-to-phoneme conversion,” in *Proc. of ICSLP*, Denver, Colorado, 2002, pp. 109–112.

[7] M. Bisani and H. Ney, “Open vocabulary speech recognition with flat hybrid models,” in *Proc. of Interspeech*, Lisbon, Portugal, 2005, pp. 725–728.

[8] A. Park and J. R. Glass, “Unsupervised word acquisition from speech using pattern discovery,” in *Proc. ICASSP*, Toulouse, France, 2006, pp. 409–412.

[9] S. Seneff, “Reversible sound-to-letter/letter-to-sound modeling based on syllable structure,” in *Proc. NAACL-HLT*, Rochester, NY, 2007, pp. 153–156.

[10] G. F. Choueiter, S. Seneff, and J. R. Glass, “Automatic lexical pronunciations generation and update,” in *Proc. of ASRU*, Kyoto, Japan, 2007, pp. 225–230.

[11] J. R. Glass, “A probabilistic framework for segment-based speech recognition,” *Computer Speech and Language*, pp. 113–127, 2003.