Faster Segmentation-Free Handwritten Chinese Text Recognition with Character Decompositions

Théodore Bluche Ronaldo Messina
A2iA SAS, Paris, France
Email: {tb,rm}@a2ia.com

***Abstract*—Recently, segmentation-free methods for handwritten Chinese text were proposed. They do not require character-level annotations to be trained, and avoid character segmentation errors at decoding time. However, segmentation-free methods need to make at least as many predictions as there are characters in the image, and often a lot more. Combined with the fact that there are many characters in Chinese, these systems are too slow to be suited for industrial applications. Inspired by the input methods for typing Chinese characters, we propose a sub-character-level recognition that achieves a 4x speedup over the baseline Multi-Dimensional Long Short-Term Memory Recurrent Neural Network (MDLSTM-RNN).**

I. Introduction

Most systems that recognize Chinese Text operate at the character level [1], [2], [3], [4], and require a pre-segmentation of the text lines. For English, French or Arabic handwritten text recognition (HTR), this kind of approach has long been abandoned, in favor of segmentation-free methods. The state-of-the-art systems for text line recognition are Multi-Dimensional Long Short-Term Memory Recurrent Neural Network (MDLSTM-RNNs [5]), winning most international evaluations in the field [6], [7], [8], [9]. Recently, MDLSTM-RNNs were applied to Chinese HTR to avoid the need of a character segmentation [10].

Although the reported results are comparable to the state-of-the-art, the authors conclude by pointing out the slowness of the system. Indeed, the main difference with Latin or Arabic HTR is the number of characters to recognize: several thousands instead of a few hundreds, leading to a very large output space, and a big output layer in the neural network. While this is not specific to segmentation-free methods, it is magnified by the fact that those approaches tend to make many more predictions than there are characters, and the subsequent decoding of the sequence of predictions allows to retrieve the character segmention.

Interestingly, handling a large alphabet is also problematic for Chinese speakers, when they want to type some text on a keyboard. Input methods were devised to tackle this issue and may be grouped in two main categories, both enabling the writer to type with a standard QWERTY keyboard. In phonetic-based input methods, such as the popular pinyin method, the user types the pronounciation, which is automatically translated into the corresponding sequence of characters. Shape-based methods take advantage of similarities of parts of characters. The characters are represented as a sequence of elementary shapes or constituants.

The existence of such methods shows that the alphabet required to produce Chinese text may effectively be reduced from several thousands symbols to less than a hundred, and that a sequence made of this alphabet can straightforwardly be converted into a sequence of Chinese characters. Moreover, in shape-based methods, the decomposition of a character corresponds to a graphical realization, which might be suited to visual character recognition. If we train a system to recognize these decompositions, we may also reasonably hope that characters that were not seen during training could still be recognized by their decomposition.

In this paper, we propose to benefit from shape-based input methods to accelerate the models, without any major modification of the baseline MDLSTM-RNN. We experimented this approach with two methods: Cangjie [11] and Wubi [12], and with an arbitrary encoding of the Chinese alphabet. We show that the MDLSTM-RNN can successfully learn to produce these decompositions, achieving an edit distance at the Cangjie or Wubi level below 5.0%. While the character error rate on Task 4 of the ICDAR 2013 Handwritten Chinese Character Recognition Competition [3] is not really competitive (17.0% vs. 10.6%), the huge speedup of a factor 4.0 makes the proposed approach fitted for industrial applications.

The remaining of this paper is organized as follows. First, in Section II, we give more details about the character decompositions applied in this work, namely Cangjie, Wubi, and arbitrary. In Section III, we present the data used to train the optical and language models. These models are then described in Section IV. We present our results in Section V, along with an analysis of the recognition times. Finally, we conclude by suggesting ways of improving the recognition accuracy.

II. Decomposition of Chinese Characters

Chinese input methods, either phonetic-based or shape-based, intend to provide Chinese writers with a way of typing as natural as possible. While we underlined the advantage of shape-based decompositions for visual recognition, those methods often fail to provide basic graphical

units that look the same in all conditions, but they ensure some similarities. In this paper, we investigated two popular methods: Cangjie and Wubi, briefly described below.

On the other hand, the main motivation for this work was to drastically reduce the size of the alphabet, betting on the ability of LSTM networks to model complex sequences and relationships within. If such a neural network is able to classify Chinese characters accurately from a set of more than 3,000 symbols, we may reasonably hope that it can also produce an internal representation from which more than 3,000 distinct sequences of symbols can be generated. From this stand point, decomposing characters on a graphical basis may not be quite necessary. Therefore, we also created arbitrary character decompositions.

The decomposition of a sequence of characters is obtained by concatenating the decompositions of each character. In this work, we added a symbol between each character decomposition, which will be predicted by the system. This trick allows to apply the system without any language model and evaluate the character error rates by simply converting all predictions between two delimiters back to characters.

A. Cangjie Input Method

The Cangjie method defines an alphabet of 24 basic graphical units and two special codes. They form an overall symbol set with 26 elements, which can be mapped to the letter keys of QWERTY keyboards. The 24 basic units, called radicals, are simple Chinese characters and may be used alone, or associated to form sequences. In this case they represent a graphical decomposition of the character into parts that are (to some extent) visually similar to the radicals. A few examples are shown in Figure 1, where we note that the same radical (*e.g. i*), may be used in different contexts to represent different shapes.

Figure 1. Examples of Cangjie decompositions (from Wikipedia [13], public domain).

A character encoding is typically obtained by combining one to five of these symbols. The symbols, or equivalently the keystrokes, are grouped into four subsets. The keys `A–G` constitute the philosophical set, and contain the characters representing the sun, moon and elements. Keys `H–N` represent small or simple strokes. Keys `O–R` form the body-related set, and `S–W` and `Y` the shapes set, representing complex character forms. The two special keys are `X`, used to disambiguate decompositions or to mark a character that is difficult to decompose, and `Z`, which is an auxiliary code to produce special characters such as punctuation.

The decomposition does not correspond to the stroke ordering, but follows some rules. The order of the decomposition is generally from left to right, top to bottom, and outside to inside. The geometrically connected characters are made of four codes. The unconnected characters are split into subforms according to the decomposition order, and represented but the successive decomposition of those subforms. Moreover, additional rules favor shorter decomposition, which can result in omission of some codes.

A list of decomposition of more than 20 000 characters is publicly available online.[1]

B. Wubi Input Method

In the Wubi (or Wubizixing) method, the keyboard is divided into five regions, each one assigned to a certain type of stroke when Chinese characters are written with a brush. Those five strokes are falling left, falling right, horizontal, vertical, and hook, and the zones that represent them are `QWERT`, `YUIOP`, `ASDFG`, `HJKLM`, and `XCVBN`. A total of 25 keys are used.

Each character is split into components that are most frequently radicals in Chinese script and the corresponding keys are typed in the same order in which the components would have been written by hand, according to the conventional way. To limit the number of keystrokes for very complex characters, the number of keys is limited to four. Characters that comprise more than four strokes are encoded by the first three components followed by the last one.

The majority of characters can be uniquely defined by the sequences of four keystrokes.

Some characters have a “short” and “full” representation. The former allows the user to type faster on a keyboard. In order to avoid have too many characters represented by the prefix of another one, the longer version was privileged in our work.

A list of decompositions is publicly available online.[2]

C. Arbitrary Encoding

In order to check whether the shape-based decomposition is important, we also defined an arbitraty encoding of the characters. We decided to fix the length of the code to be the same for each character, and selected the smallest alphabet allowing to cover the distinct characters in our training set, *i.e.* about 3,000. We tested three lengths – 2,

[1] http://bit.ly/1ObFPLM
[2] http://bit.ly/1ZNpNZp

3 and 4 – corresponding to three alphabet sizes: 52, 14 and 8 respectively. In the following, we will refer to these encodings as *Arb2*, *Arb3*, and *Arb4*.

For each length l with alphabet $\mathcal{A}$, we generated all possible words of $\mathcal{A}^l$ and randomly assigned each of the original character to a different word. With these decompositions, two characters with similar codes will not necessarily have a similar shape.

III. Databases

A. Images

We use the data from the CASIA Off-line Chinese Handwriting Database to train the "optical model" (see Section IV) of the recognition system. Only the sub-parts comprising images of lines of text were retained (HWDB2.0–2.2). The details on the collection of data, number of characters and lines are presented in [14]. and we omit them here for brevity. All these data were grouped and we randomly split it into two sets for developing the MDLSTM-RNN. About 85% is used to adjust the free parameters of the model and the remaining are used to assess if the parameter estimation should stop. The data used to assess performance ("evaluation" dataset) is the one from the ICDAR 2013 Handwritten Chinese Character Recognition Competition [3], which is a held-out part of the CASIA database. We only explore the Task 4, where lines of text in a page are given to the system, so that there is no influence of automatic line detection on the recognition performance. Our evaluation dataset is in fact a bit smaller than the reported one, as we ignored lines containing invalid characters (when reading the binary format in which the data is presented), resulting into 3 397 lines and 90 635 total characters; the number of different characters is 1 379.

B. Textual data

We use the transcriptions in the CASIA database and three openly available corpora: PH corpus [15] with newswire texts from the Xinhua News Agency between January 1990 and March 1991, the Chinese part of the multi-lingual UN corpus [16] and the Lancaster Corpus of Mandarin Chinese [17]. Constrarily to [10], we do not remove lines from these corpora, as *a priori* we could recognize any Chinese character given its decomposition and are not restricted by the output of the network (c.f. Section IV-A).

IV. Recognition system

The recognition system is made of two principal components. The *optical model* takes as input the pre-processed image, and outputs sequences of characters or more generally symbols probabilities. The *language model* comprises a dictionary and a n-gram model of character sequences, in order to process the output of the optical model and correct potential mistakes.

We applied a very simple pre-processing, consisting in a conversion of pixel values into normalized gray levels and a scaling of the image to a resolution of 300 dpi, clipping the height of the text line to 200px.

A. Optical Model

The optical model of our system is based on the Multi-Dimensional Long-Short Term Memory Recurrent Neural Network (MDLSTM-RNN [5]). The baseline has the same architecture as the model applied in [10] but with less than 3,000 character outputs instead of 7,350. The tiled image is presented to four MDLSTM layers which correspond to the four possible scanning directions, extracting four features each. 12 convolution filters of size 2×4 are then applied and the feature maps are summed. This procedure is repeated three times, with 20 and 32 features, followed by 100 and N features, where N is the number of predicted labels. The last layer is fully-connected (1×1 convolution). The N feature maps are summed across the vertical dimension. The dropout regularization [18] is applied after each LSTM layer of the network [19].

In order to enable the model to capture more the dependencies in the decompositions, we set $N = 100$ and added two Bidirectional LSTM layers after the collapse layer, with 128 LSTM cells each. A softmax is applied on top to obtain a sequence of predictions. We initialize the weights of the layers up to the last MDLSTM with those of the model of [10]. The initialization of the upper layers is random.

The network is trained with Connectionist Temporal Classification (CTC [20]), which define the output set of the RNN: one for each character, plus a *non-charater* symbol. This method sums over all possible segmentation of the prediction sequence yielding the correct labeling, to minimize the negative log-likelihood (NLL) of the sequence of characters. Contrary to [10], we did not train the model on isolated characters, and we therefore used a lot less data. The whole architecture is trained with RMSProp [21], using minibatches of 8 examples.

B. Language Model

We estimate character 3-gram Language Model (LM) on lines of text, handled as sequences of Chinese characters. Punctuation marks and digits are treated as any other character in the LM. The LM is estimated using the SRILM toolkit [22], pruned at $1e-7$ and smoothed with Witten-Bell [23] algorithm. We use the Kaldi [24] decoder to apply the language model on the network outputs, using finite-state-transducers [25] created using the popular "HCLG recipe" [26].

Using a "lexicon" (L) to transduce from the decompositions to Chinese characters is convenient as we just have to modify the lexicon transducer to change from one encoding to another, keeping the same "grammar" (G) transducer. For

example, the character 帅 is encoded as “L L L B” in Cangjie and as “j m h h” in Wubi.

V. Results

A. Decoding Time Analysis

The main motivation of the proposed approach is not to increase the performance of the recognition engine, but rather to speed it up. Therefore, we start this result section with a short analysis of the decoding time of the baseline model of [10], and of the proposed model. In Figure 2, we report the average decoding time per text line measured on a small sample of the development set, and a finer measure of times spent in the different layers.

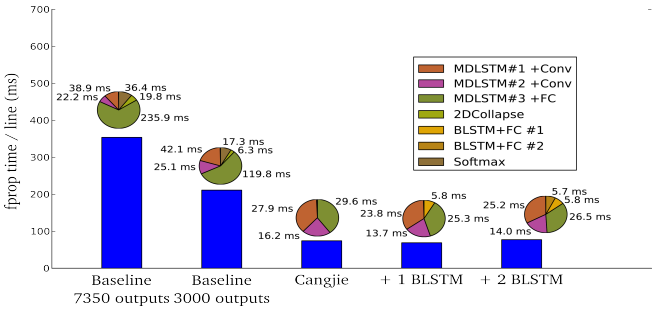

Figure 2. Total and layer-wise measures of forward propagation times. Bars indicate the average time to process a line of text. Pie charts above show the repartition of those times in different layers (layers with times $<$ 1ms are not shown).

The first noticeable fact is the huge speedup of a factor 2.8 to 4.7 with the Cangjie character decomposition. In the baseline model, about more than half of the time is spent in the topmost block of MDLSTM+fully-connected. Actually the fully-connected layer has as many output feature maps as there are characters in the alphabet. In that situation, the hundreads of 100-dimensional feature vectors coming from the previous MDLSTM layer are multiplied by a huge 100×3000 or 100×7350 matrix. The reduction of the output space from 7350 to 3000 halves this time and the reduction to a hundread dimensions yields a 4x further speedup of this layer.

Moreover, in the baseline model, and as pointed out in [10], the softmax normalization takes a significant amount of time. Note that in our implementation of the softmax layer, the shift invariance of this function is exploited to avoid numerical instability. The prediction for class k at timestep t is obtained by

$$y_k^{(t)} = \frac{e^{a_k^{(t)} - \max_n a_n^{(t)}}}{\sum_l e^{a_l^{(t)} - \max_n a_n^{(t)}}} \quad (1)$$

where $a_k^{(t)}$ are the pre-softmax activations. That computation involves finding the maximum activation among several thousands for each timestep. If we remove that shift, or if we replace it with the maximum activation across all timesteps, we obtain a 2-3x speedup of this layer.

More surprisingly, the *Collapse* layer, which only sums the feature vectors vertically, takes about 20ms. Again, it is probably due to our implementation, and the organization of feature maps in memory, which was optimized for different scenarios and is not suited to vertical sums. The computation time of this layer is however insignificant with less features.

On the other hand, adding BLSTM layers after the collapse layer to model the sequences of codes more accurately does not add much overhead (about 6ms each), compared for example to the 20+ms gain in the collapse and softmax layers.

B. Decomposition Recognition

In this section, we evaluate the ability of the model to learn the decomposition. We measured two kinds of edit distances. The edit distance between the predicted and true sequences of codes, according to the presented decomposition allows to show whether the networks can recognize sequences of codes instead of characters. On the other hand, the edit distance between sequences of Chinese characters would allow to see whether the model manages to actually decompose the Chinese characters into code sequences. In order to measure the character edit distance, we split the (predicted) code sequences on (predicted) character delimiters, and used the mappings to transform each subsequence into the corresponding character. When the code sequence did not correspond to a character, we replaced it with a special `unk` token.

Table I
Normalized edit distances (%) between codes and characters sequenceson the validation set with different approaches.

Model	Codes edit.dist	Char. edit.dist
Baseline	5.1	
Cangjie	4.5	21.7
Wubi	5.0	11.4
Arb2	8.4	15.5
Arb3	18.1	33.6
Arb4	19.2	36.6

The results are presented in Table I, on our held-out validation set. Note that code edit distances are not strictly comparable to each other because the ground-truth are different, the sequences have different lengths, and the decompositions are at a more atomic level than characters. Yet the models learnt the sequences of codes pretty accurately, with less than one in twenty wrong code for Cangjie and Wubi decompositions. Even the arbitrary decompositions are learnt, although two similar characters may have distant code sequences in these cases. This shows that the RNN can be trained to output sequences instead of classes. However, it looks clearly easier to output short sequences, despite a larger alphabet, than long ones.

An error in codes may result in an unknown character decomposition, which partly explains the high character error rates, compared to the baseline. Although Cangjie was better in code edit distance, it is largely outperformed by Wubi, showing that some decompositions may be more suited than others for handwriting recognition, and, interestingly, by the `Arb2` system, for which a code error weights less. Indeed the code length is two in that case, so one error in 12 codes is more or less equivalent to one error in six characters, while the code lengths on Cangjie is typically 4-5, so one error in twenty codes means one error in five characters.

C. Character Recognition

In this section, we add the language model and report the results on the evaluation data from Task 4 of the ICDAR 2013 Handwritten Chinese Character Recognition Competition, in Table II. The baseline is still the same system as [10], with an output space restricted to only the 3,000 characters present in the smaller training set we used. For this baseline and the arbitrary decompositions, only those characters can be recognized, so the language model was smaller and less accurate than the one applied to Cangjie and Wubi decompositions, and to [10].

Table II
CHARACTER ERROR RATE (%) ON CASIA EVALUATION DATASET, FOR OUR APPROACH WITH RAW RNN DECODING AND WITH LANGUAGE MODELS. STATE-OF-THE-ART PERFORMANCE IS ALSO INDICATED.

Decomposition	**RNN+Map**	**RNN+LM**
None (baseline)	18.0	14.6
Cangjie	32.8	17.0
Wubi	23.4	17.6
Arb2	28.0	25.7
Arb3	43.5	38.9
Arb4	46.5	40.9
[10]	**16.5**	**10.6**
[2]	**10.7**	
[3] ICDAR 2013 winner	13.3	

Although high with a simple mapping, the error rates are not too far from the baseline when we add a language model to the shape-based methods. The language model is also able to correct unknown decompositions. However, distance between similar characters, likely to be confused, may be high for arbitrary decompositions. While the results still improve with the LM for `Arb3` and `Arb4`, the character error rates become prohibitive.

Nonetheless, for Cangjie and Wubi methods, the performance loss is acceptable (+16.4% relative to the baseline), given the 2.8x speedup. For a fair comparison with [10], we still have to train our models on the same amount of data.

D. Discussion

Recognizing character decompositions instead of characters in Chinese is quite similar to recognizing characters instead of words in English. With shape-based decompositions, recognizing characters that were not seen in the training data should be possible, although we did not verify it. We left it for a future and more thorough analysis, also looking at the errors and confusions.

Moreover, we observe, to some extent, that the chosen decomposition matters. Comparing input methods with `Arb4`, yielding similar code lengths, we confirm the importance of the relation of the code to the shape of the character. We should note that any decomposition or encoding is optimized for a specific purpose. An Huffman encoding, for example, optimizes the code length. Input methods are meant to be suited for human writers. Finding an optimal character decomposition for automatic recognition would be an interesting direction of research.

However, the promising results achieved by the `Arb2` system opens an orthogonal path: why not learn unicode codepoints? Indeed, Unicode is a valid encoding of more than a million characters, with code lengths varying from one to four, and an alphabet of manageable size. Such an encoding would allow to build a universal system potentially able to recognize any language, leveraging a huge amount of training data.

VI. CONCLUSION

In an industrial context, a degradation of performance is often an acceptable price to pay to get a fast product. The model presented in [10] was very good but cripplingly slow. In this paper, we proposed a method to speed it up by a factor 4, making it almost as fast as the models for French or English. Instead of performing a segmentation-free recognition of Chinese characters directly, we trained an MDLSTM-RNN to recognize character decompositions, corresponding to different shape-based input methods, or to arbitrary encodings.

The comparisons we presented showed that the RNN could learn to output sequences instead of characters, that the chosen decomposition matters (shape-based work best, Wubi is better without a language model), and that the results stays in an acceptable range. These results are promising beyond Chinese recognition. Future work directions include learning the decomposition instead of using input methods, and recognizing Unicode codepoints directly, which would allow to build an universal, multi-lingual model.

REFERENCES

[1] S. N. Srihari, X. Yang, and G. R. Ball, "Offline Chinese handwriting recognition: an assessment of current technology," *Frontiers of Computer Science in China*, vol. 1, no. 2, pp. 137–155, 2007.

[2] Q.-F. Wang, F. Yin, and C.-L. Liu, "Handwritten Chinese Text Recognition by Integrating Multiple Contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 8, pp. 1469–1481, 2012.

[3] F. Yin, Q.-F. Wang, X.-Y. Zhang, and C.-L. Liu, "ICDAR 2013 Chinese Handwriting Recognition Competition," in *12th International Conference on Document Analysis and Recognition*, ser. ICDAR '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1464–1470.

[4] Y. Zou, K. Yu, and K. Wang, "Continuous Chinese handwriting recognition with language model," in *11th International Conference on Frontiers in Handwriting Recognition*, 2008, pp. 344–348.

[5] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in Neural Information Processing Systems*, 2008, pp. 545–552.

[6] S. Brunessaux, P. Giroux, B. Grilhères, M. Manta, M. Bodin, K. Choukri, O. Galibert, and J. Kahn, "The Maurdor Project: Improving Automatic Processing of Digital Documents," in *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*. IEEE, 2014, pp. 349–354.

[7] J. A. Sánchez, V. Romero, A. Toselli, and E. Vidal, "ICFHR 2014 HTRtS: Handwritten Text Recognition on tranScriptorium Datasets," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[8] A. Tong, M. Przybocki, V. Maergner, and H. El Abed, "NIST 2013 Open Handwriting Recognition and Translation (OpenHaRT13) Evaluation," in *11th IAPR Workshop on Document Analysis Systems (DAS2014)*, 2014.

[9] E. Grosicki and H. El-Abed, "ICDAR 2011-French handwriting recognition competition," in *International Conference on Document Analysis and Recognition (ICDAR2011)*. IEEE, 2011, pp. 1459–1463.

[10] R. Messina and J. Louradour, "Segmentation-free handwritten chinese text recognition with lstm-rnn," in *International Conference of Document Analysis and Recognition (ICDAR)*, 2015.

[11] Wikipedia. Cangjie input method. [Online]. Available: https://en.wikipedia.org/wiki/Cangjie_input_method

[12] ——. Wubi method. [Online]. Available: https://en.wikipedia.org/wiki/Wubi_method

[13] Samwingkit, online, June 2007. [Online]. Available: https://en.wikipedia.org/wiki/Cangjie_input_method

[14] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, "CASIA Online and Offline Chinese Handwriting Databases," in *ICDAR*. IEEE, 2011, pp. 37–41.

[15] G. Jin. The PH corpus. [Online]. Available: ftp://ftp.cogsci.ed.ac.uk/pub/chinese

[16] A. Eisele and Y. Chen, "Multiun: A multilingual corpus from united nation documents," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, Eds. Valletta, Malta: European Language Resources Association (ELRA), may 2010.

[17] A. McEnery and Z. Xiao, "The Lancaster Corpus of Mandarin Chinese: A Corpus for Monolingual and Contrastive Language Study," in *LREC*. European Language Resources Association, 2004.

[18] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[19] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *International Conference on Frontiers in Handwriting Recognition*, 2014.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[21] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, 2012.

[22] A. Stolcke, "SRILM - An Extensible Language Modeling Toolkit," in *Proceedings International Conference on Spoken Language Processing*, 2002, pp. 257–286.

[23] I. Witten and T. Bell, "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression," *IEEE Transactions on Information Theory*, vol. 37, no. 4, 1991.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[25] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, ser. Lecture Notes in Computer Science, vol. 4783. Springer, 2007, pp. 11–23, `http://www.openfst.org`.

[26] M. Mohri, "Finite-State Transducers in Language and Speech Processing," *Computational Linguistics*, vol. 23, pp. 269–311, 1997.