Phonotactic Language Recognition Based on Time-Gap-Weighted Lattice Kernels

Wei-Wei Liu\textsuperscript{1,2}, Wei-Qiang Zhang\textsuperscript{1}, Jia Liu\textsuperscript{1}

\textsuperscript{1}Tsinghua National Laboratory for Information Science and Technology
\textsuperscript{2}Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

Abstract

Phonotactic method for spoken language recognition (SLR) deals with permissible phone patterns and their frequencies of occurrence in a specific language. Phone recognizers followed by vector space models (PR-VSM) system is a state-of-the-art phonotactic language identification system, in which any utterance can be mapped into a supervector filled with likelihood scores of the n-gram tokens (bag-of-n-gram). However, the bag-of-n-gram language model is not good at capture the long-context co-occurrence relations due to the restriction match of the n-gram phonemes and vulnerable to the insert and delete errors induced by the frontend phone recognizer. We propose a novel approach to fill the gaps based on the use of time-gap-weighted lattice kernel (TGWLK) in this paper. The kernel is an inner product in the feature space generated by all contiguous and uncontiguous subsequences in variety length in the lattice, which are weighted by an exponentially decaying factor produced by their time gap length. The results of experiments on the NIST 2009 LRE corpus demonstrate that the proposed TGWLK shows a reduction in equal error rate (EER) than baseline system.

Index Terms: language identification, time-gap-weighted lattice kernel (TGWLK), bag-of-n-gram, vector space model (VSM)

1. Introduction

Nowadays, spoken language recognition (SLR) has become an increasingly crucial technique for many applications, such as language translation systems and search engines [1]. Without loss of generality, we can consider language recognition as a classification problem [2]. Given a set of training data and associated labels, the first step is to learn characteristics of languages from the training data, and then classify a speech utterance to the most probable language based on the language model. The problem is how to define the characteristic of the languages. In phonotactic language recognition, a classical solution is to obtain features of each language extracted from training utterances such as bag-of-n-grams. Term frequency log-likelihood ratio (TFLLR) kernel method [3] provide an efficient way of comparing the similarity of two utterances via the use of the inner product of two feature vectors of bag-of-n-grams. One competitive advantage of n-gram language models is the language-independent property. However, language recognition based on bag-of-n-grams is strict to contiguous n-gram matching, so it can not tolerate to insert, delete and substitute decoding errors. So some methods is needed to count the n-grams dynamically and efficiently.

As early as in 1997, Navratil et al. have proposed the skip-gram [4] method in language identification to solve the problem. In this method a pair of phones with one phone skipped is modeled. Then a dynamic gap n-gram matching approach for high order n-grams has been proposed [5] to generalize the skip-gram matching method. After that, a gap-weighted subsequence kernel (GWSK) [6] is introduced into language recognition. The basic idea of a GWSK is similar to that of skip-gram; however, the GWSK is theoretically better formulated. The GWSK is to counting the presence of a subsequence with a penalty related to the number of gaps interspersed within it. In this way, it has the merits of not only being robust to deletion and insertion errors but also being capable of revealing the long-context co-occurrence.

In this paper, we concentrate on investigate the application of the dynamic matching approach using time-gap-weighted lattice kernel. The kernel are based on features corresponding to occurrences of certain kind of time contiguous or non-contiguous, bounded-length-subsequences that are penalized by the time length of gaps interspersed within the subsequences in the lattices. The basic idea of a TGWLK is similar to that of GWSK except be penalized by the time length of gaps, so TGWLK is also capable of revealing the long-context co-occurrence and robust to deletion and insertion errors. Experimental results show that TGWLK outperforms PR-VSM baseline language recognition system and achieve comparable performance with GWSK system. And TGWLK and GWSK system are always mutually complementary.

The rest of the paper is organized as follows. In Section 2 we review phone recognition followed by support vector machine (PR-SVM) \cite{7} language recognition system and TFLLR kernel. Time-gap-weighted lattice kernel and its implementation are introduced in Section 3. We also discuss variants and implementation of the algorithm. Experimental setup is described in section 4. The results and discussions that TGWLK experimentally compared against the PR-VSM approach are in Section 5 followed by conclusions in Section 6.

2. PR-VSM baseline system

In this work, phone recognition followed by support vector machine (PR-SVM) \cite{7} language recognition system is employed as baseline system. The phonotactic language recognition system maps the input data $x$ to a high dimensional feature supervector as:

$$
\Phi : x \rightarrow \varphi(x).
$$

(1)
Then the supervisor \( \varphi(x) \) is sent to the SVM classifier, and a decision is made based on the output belief score of the classifier [8]. In this paper

\[
\varphi(x) = [p(d_1|\ell_x), p(d_2|\ell_x), \ldots, p(d_F|\ell_x)], \tag{2}
\]

here \( d_t = s_i, \ldots, s_{i+N−1} \) is the \( N \)-gram phoneme string, \( F = f^N \) (\( f \) is the number of the phonemes of the frontend phone recognizer). \( \ell_x \) denotes the lattice generated from data \( x \) by the frontend phone recognizer. \( p(d_t|\ell_x) \) is the observed probability of the \( N \)-gram \( d_t \) in the lattice.

In PR-SVM language recognition system an SVM is employed as the classifier, the output score is computed as following:

\[
f(\varphi(x)) = \sum_{i} a_i K_{TFLLR}(\varphi(x_i), \varphi(x_i)) + d, \tag{3}
\]

here \( \varphi(x_i) \) are support vectors. \( K_{TFLLR} \) is a TFLLR kernel computed as [3]:

\[
K_{TFLLR}(\varphi(x_i), \varphi(x_j)) = \frac{\sum p(d_{q,i}|\ell_{x_i}) \cdot p(d_{q,j}|\ell_{x_j})}{\sqrt{\sum p(d_{q,i}|\ell_{x_i}) \cdot \sqrt{\sum p(d_{q,j}|\ell_{x_j})}}, \tag{4}
\]

the \( p(d_{q,i}|\ell_{x_i}) \) is the probability of \( d_{q,i} \) across all lattices. The training stage is carried out with a one-versus-rest strategy.

3. Time-gap-weighted lattice kernel (TGWLK)

3.1. Subsequence and time-gap

Let \( \Sigma \) be a set of finite phonemes and \( \Sigma^n \) be the set of all phoneme strings with length \( n \). For a string \( d \), we denote its length by \( |d| \). \( u \) denotes a subsequence of \( d \), if there exist indices \( i = (i_1, \ldots, i_u|u|) \), with \( 1 \leq i_1 < \cdots < i_u|u| \leq |d| \), such that \( u_j = d_{i_j} \), for \( j = 1, \ldots, |u| \), or \( u = d[i] \) for short. The context length (or span) of the subsequence \( u \) in \( d \) is \( l(u) = i_{u|u|} − i_1 + 1 \) and the number of gaps is \( g(u) = l(u) \) [9].

For example, phoneme string \( d = \text{phphphphphphphphphphph} \), \( \text{ph} \) denotes a phoneme, and \( \text{ph} \) denotes the \( i \)-th phoneme in the list of phonemes into the inventory of the phone recognizer. Then \( |d| = 7 \) if \( u = \text{phphphphphph} \), then \( |u| = 3, l(u) = 6, g(u) = 6 − 3 = 3 \).

In this paper, we assume a new gap based on time. Let \( t_{i_1}, t_{i_2}, \ldots, t_{i_u} \) denote the start time of the \( i_1, i_2, \ldots, i_u \) phoneme in the string \( d \). Then \( l(u) = t_{i_{u|u|+1}} - t_{i_1}, U_t(|u|) = \sum_{t_{i_1}\leq u|u|} t_{i_{u|u|+1}} - t_{i_1}, U_t(|u|) = \sum_{t_{i_1}\leq u|u|} t_{i_{u|u|+1}} = U_t(|u|) \). For the previous example, \( l(u) = t_{17} - t_{11}, U_t(|u|) = (t_{12} - t_{11}) + (t_{13} - t_{12}) + (t_{14} - t_{13}) + (t_{15} - t_{14}) + (t_{16} - t_{15}) + (t_{17} - t_{16}) \). In another word, \( g(u) \) means the sum of the time gaps of subsequence \( u \) in the string \( d \).

3.2. TGWLK

With the definition of the time gap, time-gap-weighted lattice kernel between the utterances \( x_1 \) and \( x_2 \) can be defined as:

\[
K_{TGWLK,N}(\varphi(x_1), \varphi(x_2)) = \sum_{q=1}^{F} \sum_{i=1}^{N} \sum_{u \in \Sigma^n} \frac{\phi(u|\ell_{x_1}) \cdot \phi(u|\ell_{x_2})}{\sqrt{\phi(u|\ell_{x_1})} \sqrt{\phi(u|\ell_{x_2})}}, \tag{5}
\]

where \( d_q = s_q, \ldots, s_{q+n−1} \), \( d_r = s_r, \ldots, s_{r+n'−1} \), \( u = s_u, \ldots, s_{u+N−1} \) \( (N \leq n, N \leq n') \),

\[
\phi(u|\ell_{x_1}) = \frac{\sum_{i=1}^{F} \sum_{u = d_i[i]} \lambda_{1}(t_i|x) p(d_q|\ell_{x_1})}{\sum_{i=1}^{F} \sum_{u = d_i[i]} (\sum_{t_{m+1} \in M} (t_{m+1} - t_m)/N)}, \tag{6}
\]

where \( \lambda \) is the gap penalty parameter, \( p(t_i|x) \) is the time gap weight parameter, and \( \lambda_{1}(t_i|x) \) is the time-gap-weighted penalty.

\[
p(t_i|x) = \frac{g_i(t_i)|u|}{U_t(|u|)}, \tag{7}
\]

\( U_t(|u|) \) is the average time during time for each phoneme in subsequence \( u \) and \( g_i(t_i)|u| \) is the gap time in string \( d_q \). We define the time gap weight parameter using round up function to emphasize the time gap penalty. Then the output score of the SVM classifier is computed as:

\[
f(\varphi(x)) = \sum_{i} a_i K_{TGWLK}(\varphi(x_i), \varphi(x_i)) + d, \tag{8}
\]

where \( \varphi(x_i) \) are support vectors.

3.3. Implementation of TGWLK

The architecture of TGWLK language system is shown in Fig. 1. In this paper, we decode the input utterances using the method similar to sausage to generate phone lattices, as Fig 2. First of all, the Viterbi method is used to obtain the best phone-level segmentation for each utterance. Then the posterior probabilities of these segments to all phones in the phoneme set are calculated. At last the top-N best phones for each segment are retained. We only need to calculate the posterior probabilities, but the additional time spent on generating phone lattices is less than 10 percent of that used for 1-best phone sequence. As a result, the sausage method is much faster than the traditional forward-backward algorithm of generating phone lattices [10]. In sausages, the expected counts of \( N \)-grams can be regarded as the sum of acoustic weighted \( N \)-gram counts. It has been proven that a proper normalization method should be done on the super-vectors to achieve good language recognition performance [3].

![Figure 2: An illustration of lattice similar to sausage.](image)

As illustrated in [10], the lattices consist of a series of slots, each slot contains one or several alternative edge, and each edge is labeled with a token and corresponding posterior probability. Suppose there are \( I \) slots in the lattice \( L \), and \( J \), edges in the \( i \)-th slot. Let \( C[i,j] \) denote the token in the \( j \)-th edge of the \( i \)-th slot, and \( p[i,j] \) denote the corresponding observed posterior probability. The pseudocode is given as listed in Algorithm 1. There are deeper loops for obtaining \( n \)-gram tokens \( u \) and the expected counts \( p(d_q|\ell_{x_1}) \) in the \( n \)-fold loops.
Figure 1: Architecture of TGWLK language system.

Algorithm 1 Implementation of lattice-based GWSK feature mapping.

```
1: \( \phi(u_i|x) = 0; \)
2: for \( i_1 = 1, \ldots, \min(I - N + 1, n - N + 1) \) do
3: for \( i_2 = i_1 + 1, \ldots, \min(I - N + 2, n - N + 2) \) do
4: \( \ldots \)
5: for \( i_n = i_{N-1} + 1, \ldots, \min(I, n) \) do
6: for \( j_1 = 1, \ldots, J_{i_n} \) do
7: for \( j_2 = 1, \ldots, J_{i_{n-1}} \) do
8: \( \ldots \)
9: for \( j_n = 1, \ldots, J_{i_1} \) do
10: \( u = \sum_{i, u = d_k} \rho(d_k|\ell_x) \phi(u) + \lambda \rho(\ell_{1:N}) \); 
11: end for
12: end for
13: end for
14: \( \ldots \)
15: end for
16: end for
17: end for
18: \( \ldots \)
19: end for
20: end for
```

When \( n = I \), Algorithm 1 is untruncated TWGLK algorithm; and When \( n \leq I \), Algorithm 1 is truncated TWGLK algorithm.

### 4. Experimental setup

#### 4.1. Baseline language recognition system

In this paper a parallel PR-SVM language recognition system is used as baseline system. The first step is to tokenize speech by the means of running phone-recognizers to provide the posterior probabilities of the phone occurrences. Hungarian (HU), Czech (CZ), Russian (RU) Temporal Patterns Neural Network (TRAPs/NN) phone-recognizers that developed by the Brno University of Technology (BUT) [11] are employed in this paper. Then, we use a popular classifier LIBLINEAR [12] to classify. Finally, we use LDA-MMI algorithm [13] for score calibration and fusion.

#### 4.2. Test, training and developing dataset

The results are reported for the test trials of the National Institute of Standards and Technology Language Recognition Evaluation (NIST-LRE) 2009. The test data consists of 41793 test segments of 23 languages for 30-s, 10-s, and 3-s nominal duration test.

30000 conversations are selected from the Call-Home, Call-Friend, OGI, OHSU and VOA Corpus used for training. 22701 conversations selected from the database provided by NIST for the 2003, 2005 and 2007 LRE and VOA are used for develop database.

#### 4.3. Evaluation measures

The performance of language recognition systems is reported in terms of Equal Error Rate (EER) and average cost performance \( C_{avg} \) which is defined by NIST LRE 2009 [14] in this paper.

### 5. Experimental results and discussion

#### 5.1. Performance of HU front-end TGWLK system

We investigate the performance of TGWLK in this subsection. Besides the decaying factor \( \lambda \), we also vary the truncated context length \( n_1 \). The results are listed in Table 1 and \( N = 3 \). Note that \( n = \infty \) corresponds to the untruncated TGWLK. Table 1 shows that for each fixed \( n \), the trend with respect to \( \lambda \) is similar to the untruncated TGWLK. The EERs/Cavgs first decrease and then increase with the increasing of decaying factor \( \lambda \), and the minimum occurs at \( \lambda = 0.2 \), which has been predicted in [6]. On the other hand, for each fixed \( \lambda \), the truncated TGWLK changes to standard TGWLK with increasing \( n \). The EERs/Cavgs are monotonously increasing for large \( n \); as for proper \( \lambda \), the EERs/Cavgs are first decreasing and then increasing and the minimum occurs at \( n = 9, 9 \), as we expected. We can also observe that \( n \) has little effect on the performance, based on this result, we can select small values of \( n \) to deduce the computation cost while retaining similar performance. In the subsection, we have seen that TGWLK outperformed PR-VSM method and achieve comparable performance with GWSK. And the fusion of TGWLK and GWSK system gives better results, which means the two system can be mutually complementary.

#### 5.2. Parallel Phone Recognizer Experiments

The previous subsections has shown that TGWLK outperformed PR-VSM method. In this section, we will further validate TGWLK using HU, RU, and CZ phone recognizers as frontends. We only focus on the most challenging case, i.e., PR-VSM versus TGWLK with \( N = 3 \).

As for TGWLK, we set the parameters \( \lambda = 0.2 \) and \( n = 9 \) and use the truncated version.

The results of parallel frontends system are obtained using LDA + MMI score fusion backend [13]. Fig. 3 shows the detection error trade-off (DET) curves and the EERs and Cavgs are listed in Table 2. Table 2 shows the consistent performance improvement due to changing from PR-VSM to TGWLK for both single and parallel frontends. For the HU+RU+CZ parallel frontends and 30-s test, the Cavg decreases from 1.54%, 4.11% and 15.80% to 1.49%, 3.98%, and 15.71%.
Table 1: Performance of TWGLK \((N = 3)\), HU frontend, LRE09, 30-s test (EER and Cavg in %).

<table>
<thead>
<tr>
<th>(\lambda)</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EER</td>
<td>Cavg</td>
<td>EER</td>
<td>Cavg</td>
<td>EER</td>
</tr>
<tr>
<td>TGWLK</td>
<td>(n = 5)</td>
<td>2.39</td>
<td>2.29</td>
<td>2.30</td>
<td>2.23</td>
</tr>
<tr>
<td></td>
<td>(n = 7)</td>
<td>2.38</td>
<td>2.29</td>
<td>2.29</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>(n = 9)</td>
<td>2.37</td>
<td>2.29</td>
<td>2.28</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>(n = 11)</td>
<td>2.37</td>
<td>2.29</td>
<td>2.29</td>
<td>2.24</td>
</tr>
<tr>
<td></td>
<td>(n = 13)</td>
<td>2.38</td>
<td>2.29</td>
<td>2.29</td>
<td>2.24</td>
</tr>
<tr>
<td></td>
<td>(n = \infty)</td>
<td>2.43</td>
<td>2.37</td>
<td>2.39</td>
<td>2.28</td>
</tr>
<tr>
<td>PR-VSM</td>
<td>(n = 3)</td>
<td>2.44</td>
<td>2.37</td>
<td>2.44</td>
<td>2.37</td>
</tr>
<tr>
<td></td>
<td>(n = 7)</td>
<td>2.29</td>
<td>2.24</td>
<td>2.28</td>
<td>2.23</td>
</tr>
<tr>
<td>TGWLK (+) GWSK ((n = 9))</td>
<td>2.29</td>
<td>2.23</td>
<td>2.26</td>
<td>2.17</td>
<td>2.27</td>
</tr>
</tbody>
</table>

Table 2: Performance of baseline and TGWLK systems \((n = 9)\), NIST LRE2009, (EER/Cavg in %).

<table>
<thead>
<tr>
<th>System</th>
<th>30s</th>
<th>10s</th>
<th>3s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>HU</td>
<td>2.44/2.37</td>
<td>7.38/7.24</td>
</tr>
<tr>
<td></td>
<td>RU</td>
<td>2.21/2.00</td>
<td>6.23/6.07</td>
</tr>
<tr>
<td></td>
<td>CZ</td>
<td>3.33/3.30</td>
<td>10.03/10.07</td>
</tr>
<tr>
<td>fusion</td>
<td>1.54/1.61</td>
<td>4.11/4.00</td>
<td>15.99/15.76</td>
</tr>
<tr>
<td>TGWLK</td>
<td>HU</td>
<td>2.28/2.22</td>
<td>7.08/7.16</td>
</tr>
<tr>
<td></td>
<td>RU</td>
<td>2.15/1.92</td>
<td>5.84/5.70</td>
</tr>
<tr>
<td></td>
<td>CZ</td>
<td>3.08/3.07</td>
<td>9.86/9.82</td>
</tr>
<tr>
<td>fusion</td>
<td>1.49/1.47</td>
<td>3.98/4.03</td>
<td>15.71/15.57</td>
</tr>
</tbody>
</table>

5.3. Computational Complexity

Table 3 shows the real time (RT) factors of each part of the language recognition system. For the training stage, super vector product is the dominant computational part, so compared to PR-VSM, the computational cost increases about 1.5 times for the untruncated TGWLK, and only 8.1% for the truncated TGWLK \((n = 9, N = 3)\). For the test stage, decoding and super vector generation are the dominant computational parts and the computational cost increases about 50% for the untruncated TGWLK, with almost no increase for the truncated TGWLK \((n = 9)\).

Table 3: Comparison of real time factor for PR-VSM and TGWLK, HU frontend, LRE09, 30-s test, \(N = 3\). CPU: Xeon E5520@2.27GHz, RAM: 8GB, single thread. SV gen.: super vector generation, SV prod.: super vector product.

<table>
<thead>
<tr>
<th>method</th>
<th>(n)</th>
<th>decoding</th>
<th>SV gen.</th>
<th>SV prod.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR-VSM</td>
<td>-</td>
<td>0.11</td>
<td>(1.1 \times 10^{-4})</td>
<td>(3.7 \times 10^{-6})</td>
</tr>
<tr>
<td>TGWLK</td>
<td>9</td>
<td>0.11</td>
<td>(3.7 \times 10^{-4})</td>
<td>(4.3 \times 10^{-5})</td>
</tr>
<tr>
<td>(n = \infty)</td>
<td>6.7 \times 10^{-2}</td>
<td>9.1 \times 10^{-6}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: DET curves of PR-VSM and TGWLK, LRE09, HU+RU+CZ frontend.

6. Conclusions

In this paper, an approach of time-gap-weighted lattice kernel (TGWLK) has been presented for language recognition. TGWLK is based on time-gap-weighted matching, which is not so vulnerable to deletion and insertion errors of the frontend phone recognizer than traditional contiguous \(n\)-gram matching. The experimental results evaluated on NIST 2009 LRE task show that the relative improvements of the proposed TGWLK are 3.24%, 3.16% and 1.75% for 30s, 10s and 3s over traditional bag-of-\(n\)-gram approach respectively.

7. Acknowledgements

This project is supported by National Natural Science Foundation of China (No. 61005019, No.61273268 and No. 61370034).
8. References


