19ஆம் தமிழ் இலக்கியம் ஆரம்பம் (விளையாட்டு) - 2020
19th Tamil Internet Conference (Virtual) - 2020

பொறுப்புரை பட்டகம் கையலகவுடன் இணைந்து பதிப்பிட்டுகிறது

புத்தேசியா, பக்தாவர் பக்தகல்கவுடன் இணைந்து, பிரித்தள

பதிப்பு : 11 - 13, 2020

பொறுப்புரைக் கல்விகள்
Conference Proceedings

அனைத்துப் பொறுப்புகள் வல்லாக பக்தாளார் பக்தாதாசியியா பதிப்பிட்டுகிறது.

பதிமப்
கல்வி சுற்று கையல் விளையாட்டுப் பதிப்பிட்டு, எதிர்ம.

PATRIKA.COM
பதிமப்

91 772313 488707
பவானின் கல்பத்தோறுகை

பதினாறாம் வருடம் வரும் ஆண்டுகள்
தேவதை, மாநாடு திருப்பதிகள் தொடு
பல்பீடியும் பல்தொண்டகத்தையும்
பிரித்தாரா, அருசிக்கா

பதினும்
ஏழாம் அந்தரம் தொளி தம்பாலம் என்மிக் (ஏழா மூன்று)
அருசிக்காரின் சிதிப்பரசியம் பார்ப்பின் இரைய
சுருக்கம் தியாயபாரதம் பதிப்பின் பேப்பாரம் தியாயபாரம்
19th Tamil Internet Conference
Virtual Meet
December 11-13, 2020

CONFERENCE PAPERS

Compiled by
Dr. Vasu Renganathan
Chair, Conference Program Committee
University of Pennsylvania
Philadelphia, PA 19104 USA

Published by:
International Forum for Information Technology in Tamil (INFITT)
A non-profit Organization registered in California, USA
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Program</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Greetings</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Preserving Tamil Scripts: The Way towards their Digitization, Archival and Outreach, Vinodh Rajan</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>Enhancing Tamil Teaching &amp; Learning using Student Learning Space (SLS) and Visible Thinking Routines, Louis Isack Kumar</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>Online teaching and learning during COVID-19 pandemic period: The Mauritian experience, Sandanaluchmee Vellien</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>Verb Phrase Translation in English to Tamil Machine Translation, Vijay Sundar Ram, R and Sobha, Lalitha Dev</td>
<td>58</td>
</tr>
<tr>
<td>7</td>
<td>Named Entity Recognition in Tamil from Code Mix Social Media Text, Pattabhi, R K Rao and Sobha, Lalitha Devi</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>Balance Scale to Disambiguate Postpositions in Telugu and Tamil: An Implementation in Transfer Based Machine Translation, Parameswari Krishnamurthy</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>Complexities in Developing Tamil-Brahmi Script OCR: An Analysis, M. Monisha, V. S. Felix Enigo</td>
<td>88</td>
</tr>
<tr>
<td>10</td>
<td>தமிழ் அழிப்புத் தமிழ் இலைமை நூற்றாண்டு அடிப்பு: கலந்துமருளும்</td>
<td>102</td>
</tr>
<tr>
<td>11</td>
<td>Application of Letter Successor Varieties in Tamil Morphological Analysis, Rajan, K., V. Ramalingam and M. Ganesan.</td>
<td>108</td>
</tr>
<tr>
<td>12</td>
<td>Algorithm to Correct Missing Pulli –Signs in Printed Tamil Text, Muthiah Annamalai</td>
<td>121</td>
</tr>
<tr>
<td>ID</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>13</td>
<td>Computational Analysis of Nominal Compounds in Tamil</td>
<td>Rajendran Sankaravelayuthan</td>
</tr>
<tr>
<td>14</td>
<td>TamilNayavaani - Integrating TVA Open-Source Spellchecker with Python</td>
<td>Shrinivasan, T, Nithya Duraisamy, Ashok Ramachandran, Manickkavasakam, Arunmozhi and A. Muthiah</td>
</tr>
<tr>
<td>15</td>
<td>தமிழ்முன்னைகள் குறிப்பிட்டு விளக்களிக்கப்பட்டவை மேற்கோளின் பயன்பாடு</td>
<td>பெற்றது, பன்னாட்டு திருமண விளக்கம்</td>
</tr>
<tr>
<td>16</td>
<td>புரொமுக்கல் பார்வாணியம் கைவர், பரவல் அவர்களின் தமிழ் அடையாளம்</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Generation and Parsing of Number to Words in Tamil</td>
<td>Muthiah Annaamalai, Sathia Mahadevan</td>
</tr>
<tr>
<td>18</td>
<td>SymSpell and LSTM based Spell-Checkers for Tamil</td>
<td>Selvakumar Murugan, Tamil Arasan Bakthavatchalam, Malaikannan Sankarasubbu</td>
</tr>
<tr>
<td>19</td>
<td>A Survey on Neural Machine Translation for English-Tamil Language</td>
<td>B. Janarthanasarma, T. Uthayasanker</td>
</tr>
<tr>
<td>20</td>
<td>பொருள் - செயல் குறிப்பிட்டுக்கொள்ள விளக்கக் குறியீடுகள், த. தவிர்சிப்பு, மத.</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Speech Embedding with Segregation of Para-linguistic Information for Tamil Language: a survey, Anosha Ignatius and Uthayasanker Thayasivam</td>
<td></td>
</tr>
</tbody>
</table>
Program

Nineteenth Tamil Internet Conference
December 11, 12 and 13th - 2020
Virtual Meet
Sponsored by INFITT
and Co-Sponsored by NLPC, University of Moratuwa, Sri Lanka
and Periyar University, Salem, Tamil Nadu

http://www.tamilinternetconference.org
https://www.tamilinternetconference.org/tic2020

<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 11th</td>
<td>Inauguration:</td>
</tr>
<tr>
<td></td>
<td>MC: Ms. Sandana Vellien, Educator, Seewa Bappoo State Secondary School, Mauritius</td>
</tr>
<tr>
<td></td>
<td>Welcome Speech: Dr. Arul Veerappan, Chair, INFITT</td>
</tr>
<tr>
<td></td>
<td>Introduction, CPC: Dr. Vasu Renganathan, Chair, CPC, INFITT</td>
</tr>
<tr>
<td></td>
<td>Professor Gihan Dias, Co-Chair, University of Moratuwa</td>
</tr>
<tr>
<td></td>
<td>Inaugural Speech: Professor Ponnaikko, Founder, INFITT</td>
</tr>
<tr>
<td></td>
<td>Keynote: Professor M. A. Anandakrishnan, Founder, INFITT</td>
</tr>
<tr>
<td></td>
<td>Greetings messages:</td>
</tr>
<tr>
<td></td>
<td>Professor P. Kolandaivel, Vice-Chancellor, Periyar University, Salem</td>
</tr>
<tr>
<td></td>
<td>Professor C. Subramaniam, Former Vice-Chancellor, The Tamil University, Thanjavur.</td>
</tr>
<tr>
<td></td>
<td>Release of Conference Proceedings – Conference Program Committee and members of the Inaugural meeting.</td>
</tr>
<tr>
<td></td>
<td>Vote of Thanks: Professor K. Kalyanasundaram, Former Chair, INFITT</td>
</tr>
</tbody>
</table>

Keynote Session I (6PM to 6:30PM)

Chair: Professor E. Annamalai, University of Chicago,
Speaker: Dr. Vinodh Rajan, Germany
Paper: Preserving Tamil Scripts: The Way towards their Digitization, Archival and Outreach
### Session II (6:30 to 7:00PM)

**Chair:** Professor Appasamy Murugaiyan  
**Speaker:** Louis Isack Kumar, Bartley Secondary School, Singapore  
**Paper:** Enhancing Tamil Teaching & Learning using Student Learning Space (SLS) and Visible Thinking Routines

### Session III (7:00 to 7:30PM)

**Chair:** Prof. Rajendran Sankaravelayuthan, Amrita Vishwa Vidyapeetham, Coimbatore.  
**Speaker:** Sandana Vellien, Educator, Seewa Bappoo State Secondary School, Mauritius  
**Paper:** Online teaching and learning during COVID-19 pandemic period: The Mauritian experience

### Session IV (7:30 to 8:30PM)

**Chair:** Dr. Sobha Nair, AU-KBC Research Centre, MIT campus of Anna University, Chennai  
**Speaker:** Vijay Sundar Ram  
**Paper:** Verb Phrase Translation in English to Tamil  
**Speaker:** Pattabhi, RK  
**Paper:** Named Entity Recognition in Tamil from Code Mix.  
**Speaker:** Parameswari Krishnamurthy, Center for Applied Linguistics and Translation Studies, Hyderabad  
**Paper:** Balance scale to disambiguate Postpositions in Telugu and Tamil: An implementation in transfer based Machine Translation

---

### December 12th, Saturday between 5:00PM and 8:30PM IST

**Keynote Session I (5PM to 5:30PM)**

**Chair:** Professor Gihan Dias, NLP Center, Moratuwa University, Srilanka  
**Speakers:** T. Uthaysankar, K. Sarveswaran, R. Janarthanasarma, Aloka F., and Thanuja P., Moratuwa University, Sri Lanka  
**Topic:** Introduction to Tamil NLP research at the NLP Center, Moratuwa University
<table>
<thead>
<tr>
<th>Session II (5:30 to 6:00 PM)</th>
</tr>
</thead>
</table>
| **Chair:** Dr. L. Ramamoorthy, CIIL, Mysore  
**Speaker:** M. Monisha, V. S. Felix Enigo, SSN College of Engineering, Chennai.  
**Paper:** Complexities in Developing Tamil-Brahmi Script OCR: An analysis. |

<table>
<thead>
<tr>
<th>Session III (6:00 to 6:30PM)</th>
</tr>
</thead>
</table>
| **Chair:** Dr. Arul Veerappan, Chair, INFITT.  
**Speaker:** அகில், ஏசிதெமாழி நிவன், ஏசிஎன்  
**Paper:** தரவகவழிதமிழ் வியாகம்: சிக்கழக தீட்டுக்கொண்டு |

<table>
<thead>
<tr>
<th>Session IV (6:30 to 7:00PM)</th>
</tr>
</thead>
</table>
| **Chair:** Dr. M. Ganesan, Annamalai University  
**Speaker:** Dr. K. Rajan, Annamalai University  
**Paper:** Application of Letter Successor Varieties in Tamil Morphological Analysis. |

<table>
<thead>
<tr>
<th>Session V (7:00 to 7:30PM)</th>
</tr>
</thead>
</table>
| **Chair:** Prof. K. Kalyanasundaram, Switzerland  
**Speaker:** Muthiah Annamalai, Ezhilang  
**Paper:** Algorithm to Correct Missing Pulli – Signs in Printed Tamil Text: |

<table>
<thead>
<tr>
<th>Session VI (7:30 to 8:00PM)</th>
</tr>
</thead>
</table>
| **Chair:** Maniam, S., Singapore.  
**Speaker:** சுதீப்பர் மாராள், மதுமி பாலமுறைகள்  
**Paper:** மாணவூடன - செய்யவும் தமிழியேறுத்துக்கொண்டு.  
சுதீப்பர் மாராள், மா. புராணம், மு. பாராட்டு, நம்பு. மாதா சும்பி மே விபரிக்காது தலை - சதுரிய் பாலமுறைகள், கல்லற.  
என்வே வான்பற்பயணம் சிட்டும், மதுமி பாலமுறைகள், கல்லற. |

<table>
<thead>
<tr>
<th>Session VII (8:00 to 8:30PM)</th>
</tr>
</thead>
</table>
| **Chair:** Dr. R.T. Uthayasanker, Sri Lanka  
**Speaker:** Anosha Ignatius, University of Moratuwa  
**Paper:** Speech Embedding with Segregation of Para-linguistic Information for Tamil Language: a survey |
<table>
<thead>
<tr>
<th><strong>December 13</strong>, Sunday between 5:00PM and 8:30PM IST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keynote Session I (5PM to 5:30PM)</strong></td>
</tr>
<tr>
<td>Chair: Professor Deivasundaram, Chennai</td>
</tr>
<tr>
<td>Speaker: Rajendran Sankaravelayuthan, Amrita Vishwa Vidyapeetham, Coimbatore.</td>
</tr>
<tr>
<td>Paper: Computational Analysis of Nominal Compounds in Tamil</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Session II (5:30 to 5:55PM)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair: K. Sarveswaran, Sri Lanka</td>
</tr>
<tr>
<td>Speaker: T. Shrinivasan</td>
</tr>
<tr>
<td>Topic: Tamilnayavaani – Integrating TVA Open-Source Spellchecker with Python.</td>
</tr>
<tr>
<td>T. Shrinivasan, Nityha Duraisamy, Ashok Ramachandran, Manickkavasakam, Arunmozhi, and A. Muthiah</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Session III (5:55 to 6:20 PM)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair: Elantamil, Malaysia</td>
</tr>
<tr>
<td>Speaker: ராமவிமல, ஆசிதெமாழி &amp; பாழி, ஆசிதெமாழி &amp; பாழி, மாநானயா புது கால்ஸ்கு நல்லடனி, பாண்டுமோழி.</td>
</tr>
<tr>
<td>Paper: கலைவர் கூற்றிடை சிஹிர்பி஦்யா எம்பாராட்வா பாண்டுமோழி பாண்டுமோழி</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Session IV (6:20 to 6:45PM)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair: Rama Suganthan, Tamilnadu, India</td>
</tr>
<tr>
<td>Speaker: பரததியாகக், பரததியாகக்</td>
</tr>
<tr>
<td>Paper: மாநானயா பாண்டுமோழி கூற்று, பரததியாகக் பாண்டுமோழி.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Session V (6:45 to 7:10PM)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair: Shrinivasan, T., Chennai</td>
</tr>
<tr>
<td>Speaker: Malaikkannan Sankarasubbu, Saama Technologies AI Research Lab</td>
</tr>
<tr>
<td>Paper: SymSpell and LSTM based Spell Checkers for Tamil</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Session VI (7:10 to 7:35PM)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair: Sakthivel Ramasamy, Chennai</td>
</tr>
<tr>
<td>Speaker: Sathiya Mahadevan</td>
</tr>
<tr>
<td>Paper: Generation and Parsing of Number to Words in Tamil: Mutiah Annamalai and Sathiya Mahadevan</td>
</tr>
<tr>
<td>Session VII (7:35 to 8:00PM)</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td><strong>Chair:</strong> Muthu Annamalai, California</td>
</tr>
<tr>
<td><strong>Paper:</strong> A Survey on Neural Machine Translation for English-Tamil Language pair</td>
</tr>
<tr>
<td><strong>Speaker:</strong> B. Janarthanasarma, T. Uthayasanker, Department of Computer Science and Engineering, University of Moratuwa</td>
</tr>
<tr>
<td>{janarthanasarma.13, rtuthaya}@cse.mrt.ac.lk</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Valedictory Function (8:00 to 8:30PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome speech: Mr. Maniam, Singapore (Executive Director, INFITT)</td>
</tr>
<tr>
<td>INFITT Activities: Dr. Arul Veerappan (Chair, INFITT)</td>
</tr>
<tr>
<td>INFITT going forward: Professor M. A. Anandakrishnan, Founder, INFITT</td>
</tr>
<tr>
<td>Valedictory speech: Prof. Ponnaavaikko, Founder, INFITT</td>
</tr>
<tr>
<td>Next INFITT Co-Sponsor: Dr. G. Viswanathan, Chancellor, Vellore Institute of Technology, Vellore.</td>
</tr>
<tr>
<td>Concluding thoughts on Tamil NLP research: Professor Gihan Dias, Sri Lanka</td>
</tr>
<tr>
<td>Vote of Thanks: M. A. Ganesan, Vice Chair, INFITT</td>
</tr>
</tbody>
</table>
புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டையில் நிறுவன தினசரி, 2020

புதுக்கோட்டை�
புனிதவான நூற்றாண்டு விளக்கம் மார்ச், 2020
பக்தர் 11, 12 & 13.

புனிதவான நூற்றாண்டு விளக்கம்

புனிதவான நூற்றாண்டு விளக்கத்தையும் பதிவு செய்து வெளியேற்றிய தலைமுகப்பட பிற்கும் புனிதவான நூற்றாண்டு விளக்கத்தையும் பதிவு செய்து வெளியேற்றிய தலைமுகப்பட

காந்தி பிரார்த்தனை புனிதவான நூற்றாண்டு விளக்கம்

புனிதவான நூற்றாண்டு விளக்கம் பதிவு

புனிதவான நூற்றாண்டு விளக்கம்

புனிதவான நூற்றாண்டு விளக்கம்

புனிதவான நூற்றாண்டு விளக்கம்
भूमिका: तमिळ त्रिविद्यालय, 2020

भूमिका: 11, 12 & 13.

TIC 2020 भूमिका: तमिळ त्रिविद्यालय, 2020

भूमिका: 12

भूमिका: 13

भूमिका: 14
புதிய இராசிய வருடம் திருவனந்தபுரம், 2020

நடைமான 11, 12 & 13.

தமிழக மாநாடு தினசரி தமிழாயிரத்தில் ஒன்றியமாகக் கையற்றப்பட்டிருக்கிறது. (TACE-16) இந்த சமாதிய நடைமான் தலைப்பிறகு, ஒன்றியமாகக் கையற்றப்பட்டிருக்கிறது. தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகат் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகத் தமிழகம் கையற்றப்பட்டிருக்கிறது. (TACE-16)

13
புதுக்கோட்டை முந்தி திசையன் பாதுகாப்பு, 2020

புதுக்கோட்டை முந்தி திசையன் பாதுகாப்பு, 2020

11, 12 & 13.

14
The Tamil Internet Conference has, for almost 20 years, been a forum which promoted the use of IT in the Tamil language worldwide. The initial conferences saw multiple fragmented initiatives to bring Tamil to computers and the Internet. However, by now, the Unicode standard has become accepted, and we now see many new systems and applications each year.

The National Languages Processing Centre at the University of Moratuwa, Sri Lanka aims to make all computer systems and devices fully support the national languages of Sri Lanka. Our flagship project - the SiTa translation support system - allows documents to be easily translated to and from Tamil to both English and Sinhala. This is of immense significance to the Tamil citizens of Sri Lanka who had to get documents translated themselves. In addition, the Centre develops a number of technologies, resources, systems and applications to support Tamil computing.

As the Director of the NLP Centre, I am very happy that the TIC is -at least virtually - being held in Sri Lanka this year and look forward to an excellent conference.

Gihan Dias Ph.D.
Director, National Languages Processing Centre
University of Moratuwa
Sri Lanka
பகிரம மத்திய விளக்கம் நூற்றாண்டு பதிப்பு, 2020

முறை 11, 12 & 13.

பகிரம் மத்திய விளக்கம்

(இழைவ பொருளியலின் மற்றொன்றியம் 'A' கண் வலியெடுத்து முறை)

பகிரம் முதுகுமுறை

சொல்லை - 636 011, கிருஷ்ணா, திருநெல்வேல்

பகிரம் கல்வியும் விளக்கம்

முறைத்தொடர்

நூற்றாண்டு விளக்கம் ஆக்சென்ஸ் (INFTT) ஆக்சென்ஸ் மற்றும் கொடுமை தளமான விளக்கம் குறிப்பிட்டு கொண்டுள்ளது. இந்தக் கொடுக்குவிளக்கம் குறிப்பிட்டு கொண்டுள்ளது. இந்தக் கொடுக்குவிளக்கம் குறிப்பிட்டு கொண்டுள்ளது. ராTN ஆக்சென்ஸ் மற்றும் கொடுமை தளமான விளக்கம் குறிப்பிட்டு கொண்டுள்ளது. ராTN ஆக்சென்ஸ் மற்றும் கொடுமை தளமான விளக்கம் குறிப்பிட்டு கொண்டுள்ளது. ராTN ஆக்சென்ஸ் மற்றும் கொடுமை தளமான விளக்கம் குறிப்பிட்டு கொண்டுள்ளது.

1997 ஆம் ஆண்டில் பெரும்பான்மை ஆக்சென்ஸ் கல்வி தளம் மற்றும் 2020 ஆம் ஆண்டில் முறைக் கணத்திற்கும் முற்புறப்புக்கும் விளக்கத் தளம் குறிப்பிட்டு கொண்டுள்ளது.

இன்னுமே வீர ஆக்சென்ஸ் முறையை விளக்கம் மற்றும் இடையே முறையை விளக்கம் குறிப்பிட்டு கொண்டுள்ளது. இந்தக் கொடுக்குவிளக்கம் குறிப்பிட்டு கொண்டுள்ளது. இந்தக் கொடுக்குவிளக்கம் குறிப்பிட்டு கொண்டுள்ளது. இந்தப் பெரும்பான்மை ஆக்சென்ஸ் மற்றும் கொடுமை தளம் குறிப்பிட்டு கொண்டுள்ளது. ராTN ஆக்சென்ஸ் மற்றும் கொடுமை தளம் குறிப்பிட்டு கொண்டுள்ளது. ராTN ஆக்சென்ஸ் மற்றும் கொடுமை தளம் குறிப்பிட்டு கொண்டுள்ளது. ராTN ஆக்சென்ஸ் மற்றும் கொடுமை தளம் குறிப்பிட்டு கொண்டுள்ளது.

பகிரம் முறையை விளக்கியது குறிப்பிட்டு கொண்டுள்ளது. இந்த விளங்கியது குறிப்பிட்டு கொண்டுள்ளது. இந்த விளங்கியது குறிப்பிட்டு கொண்டுள்ளது. இந்தும் விளங்கியது குறிப்பிட்டு கொண்டுள்ளது. இந்தும் விளங்கியது குறிப்பிட்டு கொண்டுள்ளது. இந்தும் விளங்கியது குறிப்பிட்டு கொண்டுள்ளது.
பதிவுகளைக் குறிப்பிட்டு, கல்வியும் தமிழும், விளக்கம் தமிழும், அத்தகுச்சு தமிழும் வருகின்றனர் அதிக்காலம். இவ்வாறே கல்வியும், இவ்வாறே கல்வியும் வருகின்றது. வருகின்ற தமிழும் வருகின்ற தமிழுக்கு பதிவேற்றும். கனிமதிப்புமதிப்பு தமிழும் விளக்கப்பட்டத் பதியவில்லை. கல்வியும் தமிழும் வருகின்ற தமிழும் பதிவேற்றும் வருகின்றது. இங்கு வருகின்ற பதிவுகளில் வருகின்ற கல்வியை அளிப்பதால், பதியும் வருகின்ற கல்வியை பதிவூட்டும். பதிவு பதிவில் வருகின்ற தமிழும் வருகின்ற தமிழுக்கு பதிவேற்றும். என்று கூறும் பதிவு பதிப்பு பதிவில் பதிவேற்றும்.
மந்திரே. புதுக்கோட்டையில் இயற்கை ஜுனியர் மாநிலமும் கிருஷ்ணாமலர் சோதனையாளர் ஹர்ஸ்க தேவகிருஷ்ணாமலர் வருவனும் மாநிலத்துறை தலைவர் வருவனாக செயல்பட்டு வருகிறார். இவர் தமிழ் தொலைக்காட்சியில் இணைந்து நிறைந்து விளங்குகிறார். அவரும் தமிழ் திட்டமும் குறுகியாக மாநிலத்துறைத் தலைவர் வருவரும் விளக்கம் செய்யும் விளக்கமே தெரிகிறது. இவருடைய செயல்பாடுகள் தமிழ்நாட்டின் வளர்ச்சியும் கலை நெறியும் கருத்துறவுகள் மதிப்பிடப்படுகின்றன. இவருடைய வாய்ந்தவை தமிழ்நாடுக்கு முன்னேற்காக செய்யப்படுகின்றன. மாநிலத்துறை தலைவராக வருகிறார் இளமையும் விளக்கம் செய்யும் விளக்கமே ஆனால் இவருடைய வாய்ந்தவை தமிழ்நாட்டின் கலை நெறியும் கருத்துறவுகளும் மதிப்பிடப்படுகின்றன.
Preserving Tamil Scripts: 
The Way towards their Digitization, Archival and Outreach
Dr Vinodh Rajan vinodh@virtualvinodh.com

1 Introduction

The Tamil language has a written tradition that spans a little more than two millennia. In these two thousand and odd years, its writing has evolved drastically from scratching in pot shreds by potters and carvings in caves by Jain monks to being displayed in smart devices around the world. During this period, it has been written in several distinct scripts, some of which shared the label Tamil and others did not. Most of these scripts are now just distant memories and frozen reflections of the past.

These myriads of scripts are an integral part of the historical milieu of the Tamil land and are to be considered cultural artefacts that needs to be actively preserved for the future. However, most of the scripts are currently locked up as scanned images and cannot be properly used for diverse purposes. If someone wants to render a text portion in any of the scripts and discuss the text in its original context, it currently cannot be done (apart from Tamil-Brahmi & Vatteluttu) with ease. Thus, there is a need to actively preserve them by digitizing them as fonts and bring them back to life.

The article will initially explain this need to effectively preserve the scripts as digital fonts\(^1\) and the various benefits such digitization will bestow upon us. It will then look at how this preservation can be performed, outlining the steps involved, at the same time mentioning the associated challenges. It will also briefly address the outreach efforts that are required to help the public embrace these historical scripts, without pushing them solely into the realm of specialists. Finally, it will share the personal experience of the author in digitizing and popularizing the Tamil-Brahmi script as a font.

---

\(^1\) A more appropriate term to use would be \textit{typeface}. For the sake of avoiding jargon, \textit{font} is used instead
2 A Brief Overview of Tamil Scripts

The figure above shows the evolution of the various Tamil scripts from their precursors. Except for the scripts that are marked yellow, the other scripts are used to write Tamil. In fact, the language was at one-point *biscriptual* i.e. it was written in two different scripts at the same time (Siromoney, 1981). Had you hailed from the *Chera* or the *Pandya* country in the 10th century CE you would have probably used *Vatteluttu*, else you would have written using the *Pallava*-derived Tamil script. As opposed to popular belief, the modern Tamil script is not directly derived from Tamil-Brahmi but rather it is an adaptation of the *Pallava* script (Siromoney, 1980) with some borrowings from early *Vatteluttu*.

---

2 கொண்ட புத்து புத்துக்காக்குறி அதிக
புத்துப்புற பூத்தாலையாய உண்டாயினும் பல்வேறு இரண்டு தமிழ் கலையாக்குகின்றன. தமிழ் கலையான முதலாம் பாண்டிய நாட்டிலிருந்து காலமான இரண்டு தமிழ் நாட்டிலிருந்து இரண்டு பேரரசுகள் ஆண்டால் இரிழாயினும் நூற்றுக்காலம் பழங்காலான பல்வேறு சமயங்களிலிருந்து பழங்காலம் ஆண்டால் இரிழாயினும் நூற்றுக்காலம் பழங்காலான பல்வேறு சமயங்களிலிருந்து பழங்காலம் ஆண்டால் இரிழாயினும் நூற்றுக்காலம் 

---

[2 கொண்ட புத்து புத்துக்காக்குறி]
The scripts associated with the Tamil language can be roughly organized into three major labels: Tamil-Brahmi, Vattelutu and (Pallava) Tamil. The first two can be further categorized roughly as early, middle and late, corresponding to their evolution and distinct appearance. The (Pallava) Tamil script due to its longevity went through several more additional phases as shown in fig. 1. As a cosmopolitan language of groups that immigrated into the Tamil land and groups that emigrated out of it, Tamil has also been written with the following scripts: Telugu, Kannada, Thai Grantha and Arabic.

This wonderful diversity in Tamil scripts as shown in fig: 2 is what needs to be preserved for the active use of the future generation.
3 Preservation as Font

It is sometimes argued that there is no need to preserve the scripts as fonts. The scripts are already preserved as scans or photographs. If we require textual representation, the content can just be rendered in the modern script. This was indeed frequently shared by many with the author during the development of the Tamil-Brahmi font. However, this does not do justice to an artefact’s historical nature and are detrimental to the long-term archival of both the artefact and its script. In this context, the main arguments for preserving the scripts as fonts and its benefits is presented below.

3.1 Text Archival

Textual artefacts such as palm leaves and estampages of inscriptions are prone to destruction and decay. In the off chance such artefacts are destroyed, preserving the contents as digital textual editions in the original script ensures some form of survival of the writing in its intended shape and form.

Historical documents are also prone to interpretations. Particularly in Tamil, historical writing is rife with ambiguous sequences that need to be contextually interpreted. These are often done by experts to produce a readable text in the modern script as transcription. However, using this interpreted modern text for archival introduces a layer of subjective interpretation. The documents are no longer archived as they are. This is prevented by creating a diplomatic transcription that imitates the original material as much as possible. While diplomatic editions can be created with transliterations or the modern script, it also introduces a layer of bias. A proper diplomatic transcription should be in the original script. This will allow future researchers to access the document without any prior bias.

For instance, a reader may not realize that the transcription செரா is actually an interpretation of the original ambiguous sequence செசா which in all probability could also be read as செரர். Archiving in the original script preserves such sequences and removes interpretive bias. All this requires the existence of a digital font for that script.

3.2 Historical Aesthetics

Nobody can deny that the content of various inscriptions and palm leaf manuscripts are of cultural and historical importance. However, the same importance ought to be given to forms of the shapes of the letters as well. A script embodies more than the underlying
language that it is attempting to represent. It reflects aesthetics of that era. This nuance and impact of viewing an artefact in its original shape and form gets lost when it is rendered in the modern variant. It simply does not evoke the same emotion, mood or ambiguity.

Digitizing them and archiving them as they were written connects the audience to a bygone era and exposes them to a Tamil that existed centuries earlier.

Consider the following image of the famous Tamil-Brahmi inscription from Jambai.

Rendering and archiving it with the apparent reading as:

```tamil
ஸியினேதாேமாநாஞைததபாளி
```

or the intended reading as:

```tamil
ஸதியினேதாெமா,அசிஈததப\(\text{ işlet })\lli
```

does not capture the aesthetics of the inscription as much as digitizing and archiving it as:

```tamil
(Digitized text using Adinatha Tamil-Brahmi font)
```

This is particularly important to text passages, where an ancient word can be discussed in running text easily without resorting to inline images. One can now discuss about ‘Sangam’ of the Sangam period in running text with ease and comfort, all the while preserving the original aesthetics of that era.

### 3.3 Understanding Transmission

Rendering texts in ways they would have been originally written is essentially to understand how texts were transmitted over the ages. Some letters may have been similar in the original script of the scribe when it was written, and it could have caused confusion among copyists and, hence, resulted in transmission errors.

---

3 satiyaputōnēṭumānāñciitapāḷi
4 satiyaputōneṭumānāñciittapa(\text{ işlet })\lli

For instance, in Tamil-Brahmi உ /u/ and ரு /ru/ are similar and in Vatteluttu ட /ṭa/, ப /pa/ & வ /va/ are similar, even though the characters look visually distinct in the modern script.

By rendering a text in various successive scripts and simulating a transmission lineage, it is possible to speculate the changes that may have occurred in texts during the transmission while copying. This will allow researchers to guess the causes for variant readings and corruptions in texts.

The hypothetical transmission of a Tirukkural couplet in multiple scripts is shown below. In each of these transmissions, there are letters that could have been confused with similar looking letters.

5 From Siromoney (1980)
Existence of fonts for different Tamil scripts can enable philologists to render texts instantaneously in different scripts and perform experiments in simulated transmissions to study ancient works that have been copied across generations.

3.4 Interpreting Texts

Consider the following oft-cited pun verse:

rub [against it] after removing the leg of the wife (tārai) of the enemy (Vālī) of the friend (Sugrīva) of the son (Rāma) of Ten-rathas (Daśaratha).

If the kāl is to be removed from tārai, it becomes tarai i.e. floor. Essentially, the verse is suggesting rubbing against the floor. This verse would not make any sense apart from the late medieval Tamil script, where kāl denotes the mark for -ā when occurring with consonants.

Similarly, there are verses in Tamil that make sense only if we can render them in the script that they were intended. If we are to render them in the modern script, they cannot be properly understood, as some of them are tied to the script of their composition. For instance, there is a verse style called Bindumati. The 11th century Tamil grammatical compendium Yāpparunkala Virutt describes it as follows (Pillai, 1998):

When all letters are accompanied by a dot it is called Bindumati

The example verse given by the compendium does not make sense if it is rendered in the modern Tamil script.
However, if we are to render it using medieval conventions, where short vowels /e/ and /o/ get a dot above, the explanation of the compendium becomes visually obvious.

In fact, the verse makes more sense in Vatteluttu, where each syllable forms a contiguous glyph, unlike the Pallava-derived Tamil scripts where many vowel marks such as அெ and அா do not fuse.

Another instance would be the reference நகர வல்கூதியான் 12 appearing in the Tevāram verse 97:16. This is glossed as பயணமுகிகள் குரு திருட்லக்கேதி 13 (bull-flagged-one) and further expanded as பயணமுகிகள் குரு திருட்லக்கேதி (because the figure of a bull lying resembles [the letter] நால்) (Sacchidanandan, -). Again, this probably does not make sense in modern Tamil. It may make sense if we render it in Vatteluttu, where the letter resembles a lying bull.

Fig 6: நால் in Vatteluttu

A font will allow us to render texts in the script of their composition at will and interpret them better.

4. Digitizing historical scripts as fonts

The key motivation of this section is to raise questions and uncover the tasks that need to be performed, the answers and results of which will guide the way to the various Tamil scripts' effective digitization, archival and outreach. The section does not provide definitive answers, as they require further extensive research. Any digitization attempt of

11 neykoṭe nekkoṭe netkkoṭe kotkkoṭe
ceykoṭen cemponkkoṭen teṇ
12 nakara velkoṭiyān
13 īṭapakkoṭiyān
14 paṭuttirukkum vaṭiviḷ īṭapam ńa pōṅciruttaliṅ
a Tamil script in the future must be able to answer the questions in a convincing manner and justify their decisions.

4.1 Font Design

To summarize and echo section 3: A font allows users to create texts in a historic script of their choice with minimal effort and instantaneously. It allows creation of textual diplomatic editions and render the original inscriptions as texts but at the same preserving the original aesthetics.

However, this requires meticulous font design that truly captures the essence of the historic script. This, as one might expect, requires lots of research and painstaking referencing to published materials, original inscriptions and alphabet charts. It also involves standardizing all the variant forms and choosing the appropriate shapes for a particular script. Such a task involves a great deal of effort and time in terms of character design.

Naturally, the following questions arise:

- How do we standardize the forms?
  - Which forms to choose?
  - Which forms to reject?
- How do we capture variants?
  - Different fonts?
  - Font features?
- Do we create a font for every single era or a generic font that covers multiple eras?

One can safely assume that there is a need for at least half a dozen fonts that would reasonably cover all the eras and categories of historical Tamil scripts. History enthusiasts and volunteers may create a font or two but a consistent attempt to digitize all of the scripts requires a properly funded approach to compensate for the time and effort. A more pragmatic question would be:

- How can this effort be funded?
4.2 Deciding the Encoding

Even if we design the shapes of the characters for a script, the next crucial step would be how to decide on the appropriate Unicode encoding for them. The existing Tamil Unicode is very efficient in expressing the modern Tamil script and is tied to the identity and behavior of that script. But this may not always hold for the historic Tamil scripts. Not only do the script grammar and behavior change but also the shapes begin to resemble modern Tamil less and less as we move back in time. At some point, this association breaks down and it may require an encoding of its own.

- How do we encode Pallava/Chola/Nayaka Tamil scripts?
  - Do they require separate encoding?
  - Can we unify them with the current Tamil Unicode?
  - How far can we go back without breaking the current encoding model?
  - How should the existing Tamil Unicode must be modified to accommodate historic orthographies?

- How do we encode Vatteluttu?

The Tamil scripts beginning from the Chola period up to the pre-modern period are quite ambiguous. This involves pulli being almost absent during writing and the glyphs /ra/ & vowel marker -ā (kāl) having the same shape. A glyph sequence such as சரை could be read as any of the following: cerā, cērā, cor, cōr, cerar, cērar based on the context. Many such sequences are common and are open to interpretation. This needs to be appropriately encoded. It is quite important for texts from manuscripts and inscriptions for their retrieval/sorting.

- How do we transfer this inherent ambiguity into the digital realm?
  - Do we need to maintain this ambiguity in terms of encoding?
  - Can we handle this at a font level and resort to higher level protocol for retrieval/sorting?

Another aspect these scripts differ from the modern script is the presence of ligatures and abbreviations.

- Do they have their own character identity and, thus, require separate encoding?
- Can we handle this at the font level using font features such as ligatures?
  - If we handle it at the font level, how do we ensure their proper search and retrieval?
Fig 7: Some ligatures occurring in Tamil manuscripts (Srinivas et al., 2001)

The overarching question to guide all the above is:

- How do these decisions affect script identity, character identity, text retrieval and long-term archival?

4.3 Outreach

As the famous adage goes காளவிரித்துக் கொள்வாரில்லை (Even after offering, there are no takers), there is no point developing such intricate fonts if it is too difficult to use them and as a result there are not any takers. Creating a font is not enough. One needs to create an ecosystem that makes using the font easier that includes creating appropriate tools and software to make them accessible.

This involves creating associated tools such as Input Method Editors (IME) that allow people to compose text in the target script, converters to transform existing Tamil text into their historic predecessor scripts and, even learning resources for people who are interested in learning the scripts. Else, these will be restricted to just specialists and tech-savvy enthusiasts.

---

15 kaṭaivirittum koḻvārillai
5 Tamil-Brahmi as a Case-Study

A decade ago, Tamil-Brahmi as the earliest attested Tamil script, did not have any usable digital resources available. Therefore, a collaborative effort was undertaken by the author, Udhaya Shankar, Shriramana Sharma to digitize the script and preserve the forms of earliest Tamil inscriptions in the digital realm. This was also to be used to create plain text representation of the various Tamil-Brahmi inscriptions in their original script without resorting to images or transliterations.

5.1 Adinatha Tamil-Brahmi Font

The font creation was a massive undertaking that took proper research, standardization and selection of forms. Several scholars were consulted during the design phase. It was a volunteer effort that did not receive any support and, hence, took considerable time for conceptualization to release.

Fig 8: Variants of \( \omega \) /ya/ in Tamil-Brahmi

To our advantage, Brahmi (along with the Tamil-specific characters) were already encoded in Unicode. Therefore, our font mapped the characters to the Brahmi Unicode in a one-to-one fashion. However, recently several insufficiencies in the original Brahmi Unicode to display Tamil-Brahmi characters in many modern platforms were noted. Therefore, the Unicode standard had to be updated (Rajan & Sharma, 2019) to allow properly display of the Tamil-Brahmi characters in all platforms. This resulted in the addition of 6 new Tamil-specific Brahmi characters to Unicode as shown in fig. 11.

The font can be downloaded from
http://www.virtualvinodh.com/download/Adinatha-Tamil-Brahmi.zip
Fig 9: Mangulam Inscription (Mahadevan, 2003)
5.2 Jinavani

Though the Adinatha Tamil-Brahmi font was released nearly a decade ago, it had very few adopters in the years that followed. Most of the early adopters were enthusiasts. The whole notion of installing a font and typing with it to compose in Tamil-Brahmi was a hurdle that prevented wide-spread adoption. Therefore, there was need to remove this. Jinavani (http://tamiljinavani.appspot.com) was conceived an attempt to make Tamil-Brahmi and Vatteluttu accessible by providing a simple web/android application that requires neither font installation nor composition.

It provides a simplified interface that will allow people to convert exiting Tamil text into these scripts. To circumvent the local installation of fonts it also allows to download the rendering as images and share them in social media. Apart from this, it was made to be educational as well by providing flip cards and gamified methods to learn the scripts. This was a huge success as it allowed people to instantly get the forms. This allowed people to render their names, text passages and even complete books in both Tamil-Brahmi and Vatteluttu.

---

16 The display of Vatteluttu was made by possibly through e-Vatteluttu OT as designed by Dr Elmar Kniprath
Fig 12: Jinavani Home Page

Fig 13: Jinavani Gamified Learning 1

Fig 14: Jinavani Gamified Learning 2
6 Conclusion
The road to preserving all the historic scripts used to write Tamil is a long and complicated one. This requires navigating a maze of decisions and extensive research. However, one needs to rightly recognize the need and immense value for such an effort and not brush it aside as a futile exercise. These are cultural and heritage artefacts that belong to the Tamil land and care must be taken that these are properly preserved and taken to the next generation. As much as the architectural and literary contributions of Cheras, Cholas, Pallavas, Pandyas and Nayakas are adored, how they wrote is also part of the Tamil heritage. Some progress has already been made in the case of Tamil-Brahmi and Vatteluttu. However, there is much more work to be done and with proper support and research a great deal can be further accomplished.

Acknowledgements
The author acknowledges fruitful discussions with Dr Jean-Luc Chevillard and Dr Shriramana Sharma that brought to his attention the existence of Bindumati and Ṇakaravelkoṭi. Special thanks to Dr Elmar Kniprath as well for creating the e-Vatteluttu OT font based on the Velvikkudi copper plates. Udhaya Sankar must also be mentioned here for providing his WIP tool that shows variations of characters in Tamil-Brahmi.

References


Enhancing Tamil Teaching & Learning using Student Learning Space (SLS) and Visible Thinking Routines

Louis Isack Kumar,
Bartley Secondary School, Singapore

Abstract:-

This paper aims to provide an overview of Student Learning Space (SLS), an online learning portal used in all schools in Singapore. This portal aims to support students in self-directed learning at school and beyond the classroom. This paper discusses how SLS and Visible Thinking Routines are implemented in schools, how an actual lesson was conducted using the SLS and thinking routines. As an educator we prepare our students for the 21st Century, we are constantly thinking how we can design effective lessons that promote thinking and discussion, make students’ thinking visible, and provide timely feedback. This paper aims the participants to know more about Thinking Routines and the design considerations in the choice of thinking routine and technology used to promote active learning and how we can leverage on different ICT tools and repurpose them to bring about active learning. Student Learning Space (SLS) will be used as the key platform. Through this paper, participants will know how thinking routines can be used in SLS portal to engage TL students to interact in a deeper manner with the knowledge that they are taught.

Singapore Student Learning Space (SLS)

Singapore Student Learning Space (SLS) is a student-centric learning platform that enhances existing teaching and learning in schools. It is a platform that will support Teaching and Learning in all Schools. In particular, it will help empower our students to drive their own learning according to their needs and interests. By
spurring our students to take greater ownership of their learning and work collaboratively with their peers, the SLS aims to support them towards becoming responsible future-ready learners. Students are encouraged to take greater ownership of their assignments, work closely with their peers, and be self-directed, responsible learners.

The SLS is primarily for students, but has an array of benefits for teachers as well. Educators can use the collaborative features on the SLS as a basis for their own classes, and share tips on best practices and delivery methods with each other. The SLS portal also facilitates teachers’ efforts in designing a wider range of learning experiences that cater to students’ diverse learning needs. SLS is an online learning portal that allows all students to have equal access to quality curriculum-aligned resources. These resources are available to all students for major subjects from primary to pre-university level. In line with the development of 21st Century Competencies (21CC), the SLS encourages learners to be self-
directed, and allows them to personalise their learning according to their needs and interests.

The SLS also provides teachers with a range of tools to customise and create meaningful learning experiences with technology. For example, teachers have access to tools that help to make students' thinking processes visible, enabling teachers to make informed decisions and provide targeted intervention to address any gaps in understanding. The SLS also serves as a common online platform for teachers to apply, adapt and share new pedagogies by facilitating collaborations among teachers across classrooms and schools. The SLS is continually being developed in response to needs of students and teachers. Curriculum-aligned resources and system tools are continually being improved and developed in line with suggestions and feedback from teachers and students to cater to students' diverse and evolving learning needs.

**Visible Thinking Routines:**

Thinking routines are tools specifically designed to help, support and guide student’s mental processes or thinking. Thinking routines are typically short and memorable with only a few steps based on carefully crafted questions – ‘What do you see?’ ‘What do you think about that?’ ‘What does it make you wonder?’ Making thinking visible in Tamil language classrooms provided students with vivid models of what processes of good thinking looked like and showed them how their engagement and participation in their classrooms helped them get deeper knowledge and deeper understanding.

Promoting students’ thinking skills and fostering their understanding and their intellectual development might be achieved through the application of thinking routines. Most of the Secondary School Tamil Language teachers are using it in
their teaching and learning, and most of the teachers are fully aware that thinking routines form the core of visible thinking that promotes the development of students' thinking skills. Thinking routines are the cornerstones for engaging and involving students in their classroom activities and tools for teachers to follow their students' thinking processes that help them discover their knowledge, misconceptions, reasoning ability, and understanding.

Making thinking visible must be at the heart of improving teaching and learning Tamil as a second language in classrooms. With the current Singapore teaching methods, students' thinking processes is often used in classroom teaching and are well enhanced and developed. Through the implementation of thinking routines, students are expected to externalize their thoughts, ideas, beliefs, thinking processes, and deep understanding. Thinking routines are a series of questions that Tamil Language Teachers can use in their classes to lead students to steps of critical thinking. These series of questions open children’s minds to observe, think, inquire, and delve into deeper thinking processes.

Thinking routines are easy to use mini-strategies that can be repeatedly used in the classroom, across a variety of content and grade levels. Routines take on more power when they are used to support students' ongoing learning. Each routine targets a different kind of thinking and by bringing their own content; teachers integrate the routines into the fabric of their classrooms. Thinking routines operate as tools for promoting thinking. Just like with anything, it is important that educators choose the right tool for the teaching and learning. Therefore, we must first identify the kinds of thinking we are trying to elicit from our students and then select the particular thinking routine that supports that thinking. By identifying the thinking, one is trying to elicit and support in students from the beginning, an educator is able to clearly assess the responses and justifications given by students with the response. Routines help direct student
thinking and structure classroom discussion. Begin by thinking about the type of thinking you want your students to do and try to identify a thinking routine that will help bring that to the foreground.

**Interactive Thinking Tools in SLS:**

Thinking routines are intended to “support and structure students’ thinking. The steps of the routine act as natural scaffolds that can lead students' thinking to higher and more sophisticated levels” (Ritchhart, Morrison, & Church, 2011, p. 47). When teacher plan activities using SLS portal and multimedia tools, teacher was able to close the gap between their learning and engagement. The thinking routines were mostly used to enhance oral conversation, comprehension and composition to engage students in critical thinking.

**Expository writing Task in SLS portal using Thinking Routines**
Being engaged in classroom activities using Thinking routines and SLS portal, it encouraged TL students to practice Tamil language and to raise questions to deepen their knowledge. Teacher assigned an expository writing task in SLS portal using video stimulus and thinking Routines, Students were asked to broaden their perspectives, and engage in topics or issues by explaining or
providing reasons. This is something that does not come naturally for students, and they need guidance in generating and selecting ideas during the pre-writing stage. With students mapping their ideas, they are able to demonstrate the connections between ideas and how these develop. As teachers employ the routines, they are able to gain insight into the thinking done by the students, and engage students in conversations that extend and add depth to their ideas. Thinking routines heled the teacher to externalise students’ thoughts through oral or written forms like pair discussion and mind maps.

Although the implementation of thinking routines using SLS portal in Tamil language classrooms was a challenge for both teachers and learners, especially at the beginning, the actual implementation assured the positive changes that took place in their journey of learning. Teachers revealed that the implementation of thinking routine using SLS portal helped them to model and teach thinking skills in an explicit and innovative way. Students realized that learning
is not a matter of just repeating information or having a good grade in a test, but it is more based on developing thinking skills as students explore Tamil language texts.

**Students’ collaborative response in SLS Portal**

**Conclusion:-**

Through thinking routines, students were able to demonstrate and question their understanding; they were able to critically think about what was presented in their Tamil language classes. The implementation of thinking routines motivated students to participate and express themselves, even if they were not able to produce correct Tamil sentences or express their thoughts in Tamil language. Through thinking routines, Tamil Language teachers will be able to generate and enliven rich classroom environment where more students’ discussion and participation occur. Students will be able to compare, reason, analyze, justify, interpret, rationalize, deduce, create, evaluate, apply and reflect. Thinking routines could be seen as one of the methods or pedagogies that can enhance students’ engagement in Tamil language classrooms and thus cultivate a culture of critical and creative thinking. Moreover, students benefited from the
collaborative thinking by refining their understanding and perspectives while working in groups using SLS Learning Experience Template such as Discussion & Collaboration.

Mrs M. Sandanaluchmee Vellien
Educator (Secondary)
Seewa Bappoo State Secondary School
Mauritius

Abstract

Since the COVID-19 pandemic has disrupted the normal lifestyle of people across the globe, the virtual world has come to the rescue. Generally, educational institutions in Mauritius (schools, colleges and universities) are based solely on conventional methods of learning, that is relying heavily on face-to-face lectures in classroom. The sudden outbreak of COVID-19 shook the entire world. This situation challenged the education system across the globe and forced educators to shift to an online mode of teaching overnight. Mauritius was not an exception to this. Like many other countries, Mauritius too had no option but to shift to online teaching and learning. The article covers the ways Mauritius adopted the online and also the pros and cons associated with online teaching during the COVID-19 pandemic crisis.

Keywords

COVID-19, education, online learning, e-learning, blended learning.

The world has constantly been witnessing natural calamities such as floods, cyclones, earthquakes, tsunami and so on, which deeply affect one country or the other. The most recent disaster is in the form of the COVID-19 which is still spreading like a forest fire around the world. All of the schools, colleges, and universities are facing lockdowns to curb Corona virus from further spreading. Mauritius Island too has not been spared from this pandemic. Many academic institutions are seeking the help of online learning in order not to hamper the teaching and learning processes. Online teaching can be said to have served as a panacea in the time of crisis. This article will highlight the ways online teaching- learning has taken place in Mauritius during the COVID-19 pandemic period and also some pros and cons associated with e-learning modes in Mauritius.
Government effort to curb the propagation of Covid-19

Mauritius is a subtropical island located in the south western Indian Ocean, with a multi-ethnic population of about 1.3 million people. The COVID-19 pandemic was confirmed to have reached Mauritius in March 2020. On 18 March 2020, amidst a surge in fresh cases being confirmed on Mauritius, the Government of Mauritius announced that all schools and universities would be shut until further notice. The Mauritian authorities adopted strict measures to prevent the spreading of COVID-19 in Mauritius. Despite the challenges in curbing the COVID-19 pandemic, Mauritius scored a very high mark on Oxford COVID-19 Government Response Stringency Index (Oxford University, 2020). Mauritius lifted the curfew on 30 May 2020. School resumed on 1st July 2020. In part, this success was due to a prompt and consistent governmental strategy.

Strengthening online learning during school closures

During the period of confinement, it was one of the duty of the Government to maintain a continuity of teaching and learning for all through remote learning and to mitigate the immediate impact of school closures. The COVID-19 (Miscellaneous Provisions) Bill and the Quarantine Bill, was provided to facilitate digital education, hence ensuring that learning remained uninterrupted during the COVID-19 outbreak.

In May 2020, the government approved the distribution of 2500 tablet computer to children who are on the social security register.

In this context, provision was made for distance education during temporary closure of educational institutions as well as making provisions for staffs of educational institutions to produce and conduct, distance education and online learning programmes, including broadcast lessons on the television through the Mauritius Broadcasting Corporation (MBC). A series of customised educational and online programmes had been created for students. Lessons were prepared and broadcasted for the Primary schools’ students (Grade 1 to 6) and lower secondary schools’ students (Grade 7 to 9) on a daily basis. For the primary school level, some teachers were requested to help in the preparation of video contents which were broadcasted on different MBC channels. Whereas for the secondary school students, the Student Support Programme (SSP) is a portal which played a vital role to ensure continuity in learning. This portal has been created in 2018 in line with the Nine Year Schooling program (NYS). In that context, the SSP was conceived to empower students to become autonomous
through self-learning. This supplementary instruction aimed to provide all the students the opportunities to consolidate their learning.

For Lower Secondary students (Grade 7 - 9), videos for different subjects were broadcasted on the MBC TV channels and could also be accessed online through the SSP portal, on the given link: www.ssp.moemu.org. Students of upper secondary level (Grade 10 - 13) were required to indulge in remote learning via few platforms such as Google Hangouts, Skype, Google classroom, Microsoft teams, WhatsApp, and a few more, though Zoom emerged as a clear winner.

Initially, there was some kind of hesitation on the part of some educators to use Zoom platform, partly due to lack of IT skills and also due to the apprehension of security and privacy issues related to the use of Zoom platform. Later on, Zoom seemed to have gained a lot of popularity among the Mauritian educators too.

Online Teaching not as an alternative mode, but a necessity

Accordingly, the government tried their best to assist each other by sprucing their existing online platforms, apps and providing training to teachers to use these platforms to the optimum level. Online training on Office 365 and Zoom were given to secondary school educators. Microsoft platform credentials were created and sent to all secondary school educators and students of grade 10 to 13. In light of the existing collaboration between the Ministry of Education and Microsoft, the Ministry has recommended the use of Microsoft teams. However, as mentioned on the website of the Ministry of Education, teachers may use any other video conferencing or content sharing tool of their preference, provided teaching and learning is taking place during the confinement period (MOE,2020). Upskilling and motivating teachers, parents and students are some of the important measures taken by the government. In addition, in every school, two IT savvy educators were selected by the Head of school as ‘Champion Educators’ who would assist students and teachers who were having any technological issues while engaging in e-learning mode.

The Ministry of Education also requested Heads of schools to create WhatsApp group to connect to educators of their respective schools. Following this, information was being shared with educators. The latter created their WhatsApp group for their respective classes to disseminate information to their students. It could be observed that it was easier for educators to connect with the students through WhatsApp, since most students are familiar with and good at using the WhatsApp social media platform. Furthermore, since not all students possess a laptop or PC and internet facilities at their places, most of them have at least a
smarterphone in the family. Those who do not have internet facilities at home were also able to get access to internet by buying daily prepaid package at a relatively low cost.

On a personal level, it was not too difficult for me to switch to online teaching as I frequently use the blended learning method to facilitate the teaching and learning process. This was mainly done via WhatsApp. However, during the COVID-19 confinement period, this was the first time I experimented with Zoom platform for online teaching. It took a couple of days for my students to get familiar with the Zoom platform which was relatively new for them and for me. However, Zoom proved to be beneficial, in the sense that it provides easy-to-use video conferencing. During my online teaching class, the Zoom screen-share option was of significant importance as it enabled me to share my desktop with the students. I would use the screen share option when I wanted the students to focus on a specific passage that I had selected for a given lesson. Since screen sharing automatically displays onto the students’ smartphone or laptops, they could instantly read along with me. These features helped students to stay on track, even though they were not physically in the room with me. In this way, Zoom helped to make my classes more engaging and enjoyable.

**Role of Head of Private and Public Schools**

In this context, the Head of both public and private secondary schools were requested to closely oversee the implementation of the measures proposed and ensure that students were connecting with their Educators on the platform.

For the proper monitoring and administration of the online teaching programme, Head of Schools were required to prepare a report on lessons covered, based on Educators’ submissions as per format given below, and provide a weekly feedback to the Quality Assurance and Inspection Division (QAID) via email. Below is an example of a weekly feedback form which every educator had to send to their respective Head of Department at the end of each week.

**Weekly Feedback on Online Teaching and Learning**

**General Information**

<table>
<thead>
<tr>
<th>Zone:</th>
<th>Subject:</th>
</tr>
</thead>
<tbody>
<tr>
<td>School:</td>
<td>Name of Educator:</td>
</tr>
</tbody>
</table>
Now, let’s consider some of pros and cons that cropped up while indulging in online teaching - learning.

**Students accessing learning opportunities**

Online teaching has proved to have significant strength and offers unprecedented accessibility to quality education (Chauhan, 2017). Such qualities of the online types of learning save us from these tough times. It is student centred and provides a lot of flexibility in terms of time and location. The e-learning methods enable us to customise our processes and procedures based on the learners’ needs. A combination of audio, videos and text were used by educators to reach out to their students in this time of crisis. This helped to maintain human touch to the teaching and learning process. This also helped in creating a collaborative and interactive learning environment where students gave their immediate feedback, asked queries and learn in an interesting manner. The Anywhere-Anytime feature of e-learning was beneficial in crisis-like situations such as COVID-19. Technology undoubtedly offers innovative and resilient solutions to tackle disruptions and allows individuals to connect, communicate and even work virtually without the need of face-to-face interaction.

**Opportunity to collaborate with classmates and teachers**

Online learning generally has a lot of opportunities available but this time of crisis will allow online learning to boom as most educational institutions have switched to this model (Favale et al., 2020). Now, academic institutions can grasp this opportunity by making their teachers teach and students learn via online methodology. This crisis will be a new phase for online learning and will allow stakeholders to look at the positive side of e-learning. Teachers can use e-learning platform to design various flexible programs for students. It will
enhance problem-solving skills, critical thinking abilities, and adaptability among the students. Teachers can develop innovative pedagogical approaches.

**Digital training to teachers**

While online teaching is valued for the unparalleled accessibility it offers to quality education, yet there are weaknesses inherent in the use of this medium that posed problems to the success of the e-learning method. For instance, in Mauritius, during the COVID-19 confinement period, the Ministry of Education invited secondary school to register on the Ministry online portal in order to be able to get access to Microsoft Teams. Afterwards, login details were sent by the Ministry to the Head of schools so that educators could make use of the platform, but training arrived 2 weeks later. In the meantime, many educators started using other digital platforms such as Zoom, Google Classroom and WhatsApp.

While e-learning went mainstream during the pandemic, the experience of online teaching got a mixed reaction from teachers I spoke to. Some commenting that technology served as an amazing tool that support instruction during the pandemic crisis, while others were rather sceptical about the benefits that students derived from this online learning.

Given that each educator may be differently skilled when it comes to technology, this can result in inconsistent standards of teaching if the government does not place enough emphasis on training. The need for more explicit training in digital competence through professional development for teachers is of paramount importance.

**Digital divide**

The challenge of the government and educational institutions was not only finding new technology and using it but also ensuring that all students were being taken on board during this time of crisis.

On the 8th May, the government has agreed to the procurement of tablets to be provided to 2572 children under the Social Register of Mauritius (SRM) attending Grade 10 to Grade 13 classes to enable them to follow online courses dispensed by the Ministry of Education. It was then realised that numerous households do not have internet access to benefit from online teaching.
For some schools, the online teaching and learning went on smoothly, whereas for others online learning became a challenge as they did not have a device and internet accessibility to carry on with learning.

In one of her press conference, the Minister of Education said that 90% of the educators conducted online classes during the confinement period, but till date no official data was published on the number of students attending the online classes.

On 22 September, the Mauritius Telecom (MT), launched the unlimited daily package at Rs 15 which hopefully will be beneficial for students and all those who do not have internet facilities at home and wish to engage with online teaching–learning.

**Achieving social Equity**

The government in his pursuit to bring social equity do not want to neglect and ignore the economically deprived students who do not have the facilities to follow the classes delivered online. In order to prevent those children to be left further behind, the government decided to bring drastic change in the school calendar and all examination dates. Normally the Mauritian school calendar started in January and end in December and exams took place at the end of the academic year. As from 2021, the academic year will start from end of June to end of April 2022. The table below give an overview of the various change in the school calendar and the examination dates.

**The Mauritian Calendar 2020 prior to COVID-19 was as follow:**

<table>
<thead>
<tr>
<th>Term</th>
<th>Date</th>
<th>No. of weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>Grade 1/Grade 7</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Friday 10 January to Friday 3rd April 2020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All Grades (7-13)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monday 13 January to Friday 3 April 2020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>School Holidays (2 weeks)</td>
<td></td>
</tr>
<tr>
<td>2nd</td>
<td>Monday 20 April to Friday 17 July 2020</td>
<td>13</td>
</tr>
</tbody>
</table>
School holidays

3 weeks for Secondary schools.
4 weeks for primary schools.

3rd

Monday 10 August 2020 to Friday 30 October 2020 (Secondary schools)
Monday 17 August to Friday 6 November 2020 (Primary schools)

Total 37

Timing of Assessments and Examinations Prior to COVID-19

<table>
<thead>
<tr>
<th>Grade 5 Modular Assessment</th>
<th>November 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade 6 Modular Assessment</td>
<td>August 2020</td>
</tr>
<tr>
<td>Grade 6 - Primary School Achievement Certificate (PSAC) Assessment</td>
<td>November 2020</td>
</tr>
<tr>
<td>Grade 9 - National Certificate in Education (NCE) Assessment</td>
<td>October/ November 2020</td>
</tr>
</tbody>
</table>

After the confinement period, the government published a new school calendar and rescheduled all national and international examination.

New school calendar (Post COVID -19)

Primary and Secondary 2020/2021

<table>
<thead>
<tr>
<th>Term</th>
<th>Date</th>
<th>No. of weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>Grade 1/Grade 7</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Friday 10 January 2020 to Wednesday 18 March 2020</td>
<td></td>
</tr>
</tbody>
</table>
### Easter holidays
Easter holidays 23 March to 5 April 2020 (2 weeks) and Covid-19 Break

<table>
<thead>
<tr>
<th>TV broadcast lessons (Grade 1 to 9) and Online teaching and learning (Grade 10 to 13)</th>
<th>5 April 2020 to 15 June 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd Wednesday</td>
<td>01 July 2020 to Friday 16 October 2020</td>
</tr>
<tr>
<td>Mid-term break (1 week)</td>
<td>Monday 19 October 2020 to Friday 23 October 2020</td>
</tr>
<tr>
<td>Monday 26 October – Friday 27 November 2020</td>
<td></td>
</tr>
<tr>
<td>School holidays (5 weeks)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3rd Thursday</th>
<th>07 January 2021 to Friday 26 March 2021</th>
</tr>
</thead>
</table>

### Total
44 + 11 weeks online teaching

### Timing of Assessments and Examinations

#### Post COVID-19- 2020/2021

<table>
<thead>
<tr>
<th>Grade 5 Modular Assessment</th>
<th>March 2021</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade 6 Modular Assessment</td>
<td>Week 1 of Dec 2020</td>
</tr>
<tr>
<td>Grade 6 Primary School Achievement Certificate (PSAC) Assessment</td>
<td>March 2021</td>
</tr>
</tbody>
</table>
Grade 9 - National Certificate in Education (NCE) Assessment

School Certificate (SC)/Higher School Certificate (HSC) Exams Cambridge International Examinations

School calendar 2021/2022

Primary and Secondary- Friday 25 June 2021 to Friday 29 April 2022

<table>
<thead>
<tr>
<th>Term</th>
<th>Date</th>
<th>No. of weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>Grade 1/Grade 7</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Friday 25 June to Friday 10 September 2021</td>
<td></td>
</tr>
<tr>
<td></td>
<td>School Holidays (2 weeks)</td>
<td></td>
</tr>
<tr>
<td>2nd</td>
<td>Monday 27 September to Friday 03 December 2021</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>School holidays ( 5 weeks)</td>
<td></td>
</tr>
<tr>
<td>3rd</td>
<td>Monday 10 January 2022 to Friday 29 April 2022</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>End of academic year. Holidays – 30 April to Thursday 23 June 2022 – 8 Weeks</td>
<td></td>
</tr>
</tbody>
</table>

As it can be observed, following the COVID-19, Mauritius have an unusual extended school calendar of 44 weeks of face to face teaching and 11 weeks of
online teaching for the academic year 2020/2021. Normally the school calendar is not more than 37 weeks, which is divided into three terms. Till date, it is still unclear if the school calendar and examination dates will be changed forever as from 2021/2022. Changing the examination dates may have serious repercussions on the lives of many citizens of this country.

Various pedagogues, trade unionists, educators, parents and students made an appeal to the government that the SC and HSC examination to take place in 2020, with a little postponement till the beginning of December. This would have ensured that students had enough time to complete the required school programs. Unfortunately, the Ministry of Education did not respond favorably to this request. Consequently, it was considered unfair, that many students were deprived of the possibility of taking part in the November 2020 examination as private candidate, even if they desired to do so.

On the other hand, students from International fee paying schools were able to take part in international examination such as the International General Certificate in Secondary Education (IGCSE) and International Baccalaureate (IB) Exams without any postponement. The COVID-19 confinement did not pose any problem in their school calendar and timing of examination and assessments. During a press conference a member of the opposition party deplored this policy decision of the government which he termed in French ‘une éducation à deux vitesses’ (a two-tier system of education) (Topfm, 2020), whereby students from affluent families studying in international fee paying schools are at an advantage as compared to those students studying in public and private schools in Mauritius. Students from those fee paying international schools will be ahead of those studying in the public and private schools in the sense that they will have the opportunity to enter college, university and the job market before students of same age group studying in public and private schools.

According to the administrator of the International fee paying private school, the period of confinement did not affect their private establishment. “We had already prepared for the confinement. We followed what was going on in the world and we knew that Mauritius was not going to be spared. We therefore implemented online teaching. Our students had resumed classes the day after confinement, which means that our students were not penalized by the confinement”, says a source within the administration of Le Bocage International School (ionnews,2020).
Furthermore, the fact that those students and teachers were already familiar with the blended learning mode which is part of the school curriculum, the availability of technological devices, resources and internet accessibility available to all contributed largely towards the success of online teaching for those schools.

**Conclusion**

COVID-19 has its impact on all segments on life. It has also strongly affected the whole education system and also introduced us to a digital aged learning. This pandemic makes us realise that the online education has its pros and cons. The question that one may ask is whether the online education during the COVID-19 outbreak was a challenge, a boon or curse for Mauritian Educational system. Whatever it is, a change of mindset among stakeholders is important to understand and accept the alternate mode of learning as the new way of instruction.

**Works Cited**


Abstract. Machine translation (MT) is one of the most active research areas in Natural Language Processing. Different approaches are being followed for this task. The characteristics of Verb phrase vary between languages, even within the languages in the same language family. This poses a challenge in translation between languages. The verb phrases include finite verb, non-finite verb, auxiliary verb, main verb, verbal particles and negation verb constructions. Verb phrases also carry information namely, tense, aspect, modal (TAM), and PNG (person, number and gender) other than the main verb. We have presented a methodology to transfer verb phrase between English to Tamil using Finite State Transducers. We have tested the methodology using VPT-IL 2018 FIRE Shared task. The results are encouraging.

Keywords: Verb Phrase Translation, Finite State Transducer (FST), Tense Aspect, Model (TAM)

1 Introduction

One of the most active research areas in Natural Language Processing across the globe is Machine Translation (MT). MT in Indian languages has picked-up in the last two decades. In developing a machine translation system, translation of the verb phrase from source language to target language is a challenging task. The verb phrases include finite verb, non-finite verb, auxiliary verb, main verb, verbal particles and negation verb constructions. Verb phrases also carry information namely, tense, aspect, modal, and PNG (person, number and gender) other than the main verb. The characteristics of the verbs vary between languages. In languages such as Tamil, Telugu, Kannada, Hindi, the subject and the finite verb of the sentence agree in PNG. In languages such as English and Malayalam, there is no agreement between the subject and finite verb. Languages vary in structure also such as SVO, SOV. These characteristics make the translation of Verb phrases from one language to another a difficult task. Machine translation systems are built using different methodologies such as rule-based, Statistical Machine translation (SMT) and Neural Machine Translation (NMT). In all these methodologies also verb phrase translation requires special attention. Consider table 1 which contains English to Tamil translation for a set of sentences.
Table 1. Sample Translation

<table>
<thead>
<tr>
<th>S.N o</th>
<th>English Sentence</th>
<th>Tamil translation using Google Translation</th>
<th>Tamil translation using Bing Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>He has been writing.</td>
<td>அவர் எழுந்திருக்கிறார்.</td>
<td>அவர் எழுந்திருக்கிறார்.</td>
</tr>
<tr>
<td>2</td>
<td>He may be writing.</td>
<td>அவர் எழுந்திருக்கிறார்.</td>
<td>அவர் எழுந்திருக்கிறார்.</td>
</tr>
</tbody>
</table>

The translations from Google and Bing in table 1 clearly points the need for accurate verb phrase translation. Google translation has given same translation for both ‘has been writing’ and ‘may be writing’ as ‘எழுந்திருக்கிறார்’. Whereas in Bing translation, ‘progressive marker’ என்றால் has not occurred in the translations. But it has given two different translations. Sobha et al. (1) has presented a work on verb phrase transfer between Tamil to Hindi, where they have explained the requirement of different types of verb transfers. In this paper, we have presented a methodology for verb phrase transfer between English and Tamil. We have used VPT-IL - 2018 at FIRE 2018 corpus in this work.

VPT-IL - 2018 shared task at FIRE 2018 (2) had the objective to boost the research in Machine translation in Indian languages. It had two tracks, translation of Verb Phrases from English to Tamil and Hindi to Tamil. There were three teams in the shared task and two teams participated in both English to Tamil and Hindi to Tamil translation. One team participated in English to Tamil translation only. Two teams have used Neural Machine Translation (NMT) and the third team has used a rule-based approach with feature derived from dependency parser’s output. The results obtained in both the translation tasks were poor. In the analysis of the results of the submissions, it has been found that the systems have failed in generating correct tense, aspect and modal (TAM) and in choosing the correct lexical substitution for the root word. In the following section we have described the dataset provided in the shared task.

1.1 Data Set

They have provided the training and testing dataset. The training data had a set of three files for each translation pair, which contains source language sentence, target language translated sentence and verb phrase mapping index. Both the source and target language translated sentences has sentence indexing. Verb phrase (VP) mapping index file has the sentence index and the information of the
position of the VP in the source language sentence and the position of the corresponding verb phrase in the target language translated sentence. The structure of the VP map index file is as follows.

```xml
<vpInfo sentId='' srcLang='en/hi' tgtLang='ta' vpId='verbphrase-id' vp_src_info=''
vp_tgt_info=''>

where:
- sentId: is the sentence Id.
- srcLang: Source language code. It can be 'en/hi'
- tgtLang: Target language code. It is 'ta' as Tamil is the target language in both the pairs.
- vpId: Each verb phrase is marked with an unique id.
- vp_src_info: 'verb phrase start position and its length'
- vp_tgt_info: 'verb phrase start position and its length'

A sample source sentence, target translated sentence and its verb phrase map index from English to Tamil is presented below in example 1.

**Ex 1:**

**English to Tamil translated Sentence:**

**Source Sentence:**

```xml
<Sent Id=24 lang='en'> Vandiyathevan did not get up .</Sent>
```

**Translated Sentence:**

```xml
<Sent Id=24 lang='ta'> வந்தியேதவன் என் கவிஞ் என் கவிஞ் கைல</Sent>
```

**VP Map Index:**

```xml
<vpInfo sentId='24' srcLang='en' tgtLang='ta' vpId='36' vp_src_info='15,14'
vp_tgt_info='16,18'>
```

The source English sentence has a verb phrase 'did not get up'. And the translated Tamil sentence has the equivalent verb phrase 'என் கவிஞ் கைல'. vp_src_info in VP Map Index has the starting and length information of the VP in the source sentence, 15 and 14 respectively. Similarly vp_tgt_info has the starting and length information of the VP in the source sentence, 16 and 18 respectively. A unique Id is given to the verb phrase. Statistics of data provided for English to Tamil verb phrase translation task is given in the table 2.
In the following section, we have described our approach to Verb phrase translation from English to Tamil.

2 Our Approach

We have used Finite State Transducer technique to transfer the verb phrases from English to Tamil. Finite State Transducers (FST) are used to stimulate a sequential logic, to recognize patterns and produce a desired sequential output.

Formally, a finite transducer \( T \) is a 6-tuple \((Q, \Sigma, \Gamma, I, F, \delta)\) such that:

- \( Q \) is a finite set, the set of states;
- \( \Sigma \) is a finite set, called the input alphabet;
- \( \Gamma \) is a finite set, called the output alphabet;
- \( I \) is a subset of \( Q \), the set of initial states;
- \( F \) is a subset of \( Q \), the set of final states; and
- \( \delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times Q \) (where \( \epsilon \) is the empty string) is the transition relation.

2.1 Verb Phrase Finite State Transducer

We present the verb phrase transfer rules using Finite State Transducer. FST are presented as State Tables for the utilization in computers. Using the verb phrase transfer rules, we have prepared the state table considering each word in a verb phrase as an input symbol to the sequentially modeled FST. Here in English to Tamil VP translation, we reverse the order of the words in the English VP, so that it can be mapped directly to Tamil. Consider the example given below.

Ex 2:
‘has been eating’

The verb phrase in example 2, will be presented as ‘eating been has’. (V+Cont Been has)
Transfer Rules for 'has_been_V+Cont' is as follows:

\['has\_been\_V+Cont'\] => \['V+Cont\_been\_has'\] => V+\(\)\(\)\(\)\+\+PNG

It can be represented in a FST as in fig 1.

![Sample FST](image)

**Figure 1: Sample FST**

State table representation of the rule in example 1 is as shown in table 3.

<table>
<thead>
<tr>
<th>Initial State</th>
<th>Next State</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>V+Cont</td>
<td>V+()()()+PNG</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>been</td>
<td>()()()+PNG</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>has</td>
<td>()()+PNG</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Here 'V+Cont', 'been' and 'has' are the input symbols to the FST. It starts from state 0 and changes to state 1 on receiving the input symbol 'V+Cont' and produces the output 'V+\(\)\(\)\(\)\+PNG', then it move to state 2, when input symbol 'been' is received and produced an output '\(\)\(\)(\)\+PNG' and finally when it receives the input symbol 'has', it moves to the end state 3, producing the output '\(\)\(\)\+PNG'.

Similarly we prepare the state table for all the verb phrase transfer rules. This forms a non-deterministic FST (NDFST) of verb phrases. The NDFST is minimized and determinised to form a deterministic FST using AT&T tool kit.

The input sentences are first preprocessed with syntactic workbench where the input sentence is enriched with morphological analysis, POS and Chunk information. The verb chunk in the preprocessed sentence is feed to the FST to generate the required target language verb phrase.

### 3 Experiment, Evaluation and Discussion

We have used 1992 sentences containing 2267 verb phrases provided in the training dataset of the shared task. The input sentences both English and Tamil sentences were preprocessed with morphological analyser for Tamil and
lemmatizer for English, followed by POS tagger and Chunker. The verb phrases marked in the training data were separated out and used to build non-deterministic FST. It is then processed with AT&T tool kit to convert it into deterministic FST. This is tested with the test dataset containing 1000 sentences and 1869 verb phrases. The testing data is also processed with syntactic processing modules as performed for the training data set. The verb phrases annotated in the English sentences are extracted and fed to the FST engine to generate the translation of the verb phrase in the target language.

The verb phrase translations were evaluated based on the five criteria given in the shared task. The five criteria for scoring is giving in the table 4.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Criteria</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Completely Correct</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>TAM and PNG Correct</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Correct root and TAM partially correct</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Correct root and wrong TAM</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Completely Incorrect</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5 shows the number of translated verb phrases under each of the criterion.

<table>
<thead>
<tr>
<th>Criteria Score</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>462</td>
<td>568</td>
<td>219</td>
<td>216</td>
<td>404</td>
</tr>
</tbody>
</table>

Precision and recall scores are presented in the table 6

<table>
<thead>
<tr>
<th>S.No</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55.92%</td>
<td>78.38%</td>
</tr>
</tbody>
</table>

The results show that the number of correctly identified TAM is higher and it has led to better performance scores. The number of unattempted verb phrases are significant. This clearly presents the lacuna of FST technique where the unseen TAMs were not handled. As it can identify only the known patterns used in building the FST, unknown input sequences were not attempted. This requires periodic improvement of the state table with unhandled set of verb phrases. Root word dictionary has to be exhaustive to substitute the correct root verbs in the target language.
Conclusion

We have presented an approach to handle verb phrase transfer, which is one of the most crucial modules in a machine translation system. Verb phrase vary between languages and it poses challenge to the translation. We have discussed a methodology for verb phrase transfer from English to Tamil using Finite State Transducers (FST). We have trained and tested the methodology with the dataset provided in VPT-IL 2018 shared task. The methodology proves to have better precision and the recall is directly proportional to the unique verb phrases in the training data.

References

Named Entity Recognition in Tamil from Code Mix Social Media Text
Pattabhi, R K Rao and Sobha, Lalitha Devi

Abstract. The penetration of smart devices such as mobile phones, tabs has significantly changed the way people communicate. This has led to the growth of usage of social media tools such as twitter, facebook chats for communication. This has led to development of new challenges and perspectives in the language technologies research. The biggest challenge in social media text is code mixing. This paper presents our work on Named Entity Recognition (NER) from the Tweets which have Tamil – English code-mix. In this we describe how the corpus collection and annotation is done. NE system is developed using Conditional Random Fields (CRFs). We have obtained F-measure of 70.93% comparable with the state-of-the-art.

Keywords: Named Entity Recognition (NER), Twitter data, Code mix data, Tamil-English Code mix, Machine Learning, Conditional Random Fields (CRFs)

1 Introduction

In the last decade, Indian language content and especially Tamil on various media types such as websites, blogs, email, chats have increased significantly. And it is observed that with the advent of smart phones more people are using social media such as twitter, facebook to comment on people, products, services, organizations, governments. Thus, we see content growth is driven by people from non-metros and small cities who are mostly comfortable in their own mother tongue rather than English. The growth of Indian language content is expected to increase by more than 70% every year. Hence there is a great need to process this huge data automatically. Especially companies are interested to ascertain public view on their products and processes. This requires natural language processing software systems which recognizes the entities or the associations of them or relation between them. Hence an automatic Entity extraction system is required.

Entity extraction has been actively researched for over 20 years. Most of the research has, however, been focused on resource rich languages, such as English, French and Spanish. The scope of this work covers the task of named entity recognition in social media text (twitter data) for Indian languages. In the past there were events such as Workshop on NER for South and South East Asian Languages (NER-SSEA, 2008), Workshop on South and South East Asian Natural Language Processing (SANLP, 2010 & 2011) conducted to bring various research works on NER being done on a single platform. NERIL tracks at FIRE (Forum for Information Retrieval and Evaluation) in 2013, 2014 have contributed to the development of benchmark data and boosted the research towards NER for Indian languages. All these efforts were using texts from newswire data. The user generated texts such as twitter and facebook texts are diverse and noisy. These texts contain non-standard spellings and abbreviations, unreliable punctuation styles. Apart from these writing style and language challenges, another challenge is concept drift (Dredze et al., 2010; Fromreide et al., 2014); the distribution of language and topics on Twitter and Facebook is constantly shifting, thus leading to performance degradation of NLP tools over time.

Some of the main issues in handling of social media texts such as Tweets are i) Spelling errors ii) Abbreviated new language vocabulary such as “gr8” for great
iii) use of symbols such as emoticons/emojis iv) use of meta tags and hash tags v) Code mixing.

For example:

*Ta:* Stamp veliyittu ivaga ativaangi .....  
*En:* stamp released these_people get_beaten ....

*Ta:* othavaangi .... kadasiya <loc>kovai</loc>  
*En:* get_slapped ... at_end kovai

*Ta:* pooyi pallakaatti kuththu vaangiyaachchu.  
*En:* gone show_tooth punch got

(“They released stamp, got slapping and beating ... at the end reached Kovai and got punched on the face”)

This example is a Tamil tweet where it is written in a particular dialect and also has usage of English words.

The research in analyzing the social media data is taken up in English through various shared tasks. Language identification in tweets (tweetLID) shared task held at SEPLN 2014 had the task of identifying the tweets from six different languages. SemEval 2013, 2014 and 2015 held as shared task track where sentiment analysis in tweets were focused. They conducted two sub-tasks namely, contextual polarity disambiguation and message polarity classification. In Indian languages, Amitav et al (2015) had organized a shared task titled 'Sentiment Analysis in Indian languages' as a part of MIKE 2015, where sentiment analysis in tweets is done for tweets in Hindi, Bengali and Tamil language.

Named Entity recognition was explored in twitter through shared task organized by Microsoft as part of 2015 ACL-IJCNLP, a shared task on noisy user-generated text, where they had two sub-tasks namely, twitter text normalization and named entity recognition for English.

The ESM-IL track at FIRE 2015 was the first one to come up with the entity annotated benchmark data for the social media text, where the data was in idealistic scenario, where users use only one language. But nowadays we observe that users use code mixing even in writing in the social media platforms. Thus, there is a need to develop systems that focus on social media texts. There have been other efforts on the code mix social media text in the applications of information retrieval (MSIR tracks at FIRE 2015and 2016). The CMEE-IL track at
FIRE 2016 came up with the entity annotated benchmark for code mix twitter data. The paper is further organized as follows: The next section describes the challenges in named entity extraction from social media texts and in particular Twitter data. Section 3 describes corpus collection, annotation and statistics. Section 4 describes system development methodology. And section 5 discusses results.

2 Challenges in Social Media Named Entity Recognition

The challenges in the development of entity extraction systems for Indian languages from social media text arise due to several factors. One of the main factors being there is no annotated data available for any of the Indian languages, though the earlier initiatives have been concentrated on newswire text. We also find that development of automatic named entity recognition systems for twitter kind of data is difficult due to following reasons:

i) Tweets contain a huge range of distinct named entity types. Almost all these types (except for People and Locations) are relatively infrequent, so even a large sample of manually annotated tweets will contain very few training examples.

ii) Twitter has a 140-character limit; thus, tweets often lack sufficient context to determine an entity’s type without the aid of background or world knowledge.

iii) In comparison with English, Indian Languages have more dialectal variations. These dialects are mainly influenced by different regions and communities.

iv) Indian Language tweets are multilingual in nature and predominantly contain English words.

3 Corpus Development and Annotation

The corpus was collected using the twitter API in different time periods. As explained in the above sections, in the twitter data we observe concept drift. Thus to evaluate how the systems handle concept drift we had collected data in two different time periods. Table 1 below shows the corpus statistics.

<table>
<thead>
<tr>
<th>Language</th>
<th>No. of Tweets</th>
<th>No. of NEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tamil-English</td>
<td>4576</td>
<td>2454</td>
</tr>
</tbody>
</table>

The corpus was annotated manually by trained experts. Named Entity Recognition task requires entities mentioned in the document to be detected,
their sense to be disambiguated, select the attributes to be assigned to the entity and represent it with a tag. Defining the tag set is a very important aspect in this work. The tag set chosen should be such that it covers major classes or categories of entities. The tag set defined should be such that it could be used at both coarse- and fine-grained level depending on the application. Hence a hierarchical tag set will be the suitable one. Though we find that in most of the works Automatic Content Extraction (ACE) NE tag set has been used, in our work we have used a different tag set. The ACE Tag set is fine grained is towards defense/security domain. Here we have used Government of India standardized tag set which is more generic. The tag set is a hierarchical tag set. This Hierarchical tag set was developed at AU-KBC Research Centre, and standardized by the Ministry of Communications and Information Technology, Govt. of India. In this tag set, named entity hierarchy is divided into three major classes; Entity Name, Time and Numerical expressions. The Name hierarchy has eleven attributes. Numeral Expression and time have four and three attributes respectively. Person, organization, Location, Facilities, Cuisines, Locomotives, Artifact, Entertainment, Organisms, Plants and Diseases are the eleven types of Named entities. Numerical expressions are categorized as Distance, Money, Quantity and Count. Time, Year, Month, Date, Day, Period and Special day are considered as Time expressions. The tag set consists of three level hierarchies. The top level (or 1st level) hierarchy has 22 tags, the second level has 49 tags and third level has 31 tags. Hence a total of 102 tags are available in this schema.

3.1 Data Format

The data with annotation markup is kept in a separate file called annotation file. The raw tweets as downloaded using the twitter API are kept as it is. The annotation file is a column format file, where each column was tab space separated. It consisted of the following columns:

i) Tweet_ID
ii) User_Id
iii) NE_TAG
iv) NE raw string
v) NE Start_Index
vi) NE_Length

For example:

Tweet_ID:123456789012345678
User_Id:1234567890
Index column is the starting character position of the NE calculated for each tweet and the count starts from ‘0’.

4 Our Methodology

Named Entity Recognition (NER) is defined as the process of automatic identification of proper nouns and classifies the identified entities into predefined categories such as person, location, organization, facilities, products, temporal or numeric expressions etc. Even though named entity recognition is a well-established research field and lot of research works are available for various languages, not much work has been done towards identification of NEs in code-mix social media text. The system architecture is shown in the below figure 1.

![System Architecture - Process Flow Diagram](image)

**Fig 1. System Architecture – Process Flow Diagram**

We analyzed the corpus to arrive at the most suitable word level features for identifying the NE which can be used for machine learning purposes. In Tamil we have POS tagger available but it is trained on Newswire text and not suitable for our task here. Since it is not suitable for our text, we have not used any syntactic processing in this work. We have used only statistical suffixes as features. And we have taken a window of three words for the training. The NER engine is developed using Conditional Random Fields (CRFs), a machine learning technique.
Conditional random fields (CRFs) are a probabilistic framework which is suitable for sequence prediction problem. It selects the label sequence \( y \) which maximizes the conditional probability of \( p(y|x) \) to the observation sequence \( x \). The probability of a label sequence \( y \) given an observation sequence \( x \) is given below

\[
z(x) = \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j f_j (y_{i-1}, y_i, x, i)
\]

Where \( x \) is the data sequence to be labelled and \( y \) is the label sequence. For example \( x \) is the range over sentences and \( y \) is the range over named entity tag, \( z \) is normalization factor, \( f_j (y_{i-1}, y_i, x, i) \) is a state transition feature function of an observation sequence and the labels at position \( i \) and \( i - 1 \). For example, our objective is to assigning the named entity tag or label \( y \) "LOCATION" to the sentence \( x \) "He is in Finland", then the transition function \( f_j (y_{i-1}, y_i, x, i) = 1 \) if \( y_i = "LOCATION" \) and the suffix of \( i \)th word is "land"; otherwise 0; If the weight \( \lambda_j \) associated with the above feature is large and positive, then the words ending with the suffix "land" are labelled as NE type "LOCATION" (Lafferty, 2001; Wallach, 2004).

We have used CRF++ toolkit for this work. The system can learn the NE patterns in the training data with the help of features provided and the language model is generated. Named entities are identified in the test data by the named entity model file.

5 Results and Discussion

Entities share common prefixes and suffixes for a particular type of Named entity. For example, the words ending with “Ur” most likely denotes the location name such as “porur”, “thanjavur” in Tamil. Hence, we consider bigrams and trigrams of prefix and suffix information as features. The features applied here are frequency based and is generic. Though the corpus is tagged with 3 level hierarchical tag set, we have only used the first level tags consisting of 22 tags and developed the NER engine in this work.

We performed a 10-fold experiment and obtained an average precision of 78.56% and a recall of 64.66%, which is comparable with the state of the art for NER in code mix social media text. Table 2 gives results summary.

<table>
<thead>
<tr>
<th>Language</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tamil-English</td>
<td>78.56 %</td>
<td>64.66%</td>
<td>70.93%</td>
</tr>
</tbody>
</table>

Table 2. System Results – Average Scores
Some of the main errors we found were as follows:
1) Named entities occurring in adjacent positions are tagged as single entity (two NEs combined as one NE)
2) one named entity with multiple tokens is tagged as two entities (Single NE split as two NEs)
3) NE boundary is not identified properly, beginning of an entity is tagged by intermediate tag, part of an entity is tagged by the system (as BIO format of tagging is followed by the system).
4) Organization names are not identified correctly.

6 Conclusion

We have presented a named entity system which can be used for identifying named entities in a code mix twitter data of Tamil – English. In our method, we have used generic statistical suffixes feature. The results obtained show that our feature is well suited. Error analysis shows there is need to include syntactic features of POS tagger and chunker to overcome boundary problems. In future, we plan to extend this work towards the development of POS tagger and Chunker suitable for Social media text.

References

7. Rajeev Sangal and M. G. Abbas Malik. 2011. Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (SANLP)


Balance Scale to Disambiguating Postpositions in Telugu and Tamil: An Implementation in Transfer Based Machine Translation

Parameswari Krishnamurthy
Centre for Applied Linguistics and Translation Studies
University of Hyderabad

1. Introduction
Postpositions are independent words which signal the marking of case related functions. Historically, most of the postpositions in Telugu and Tamil are grammaticalized forms of verbs and nouns. They lost their autonomous status as words and are frozen as postpositions (Cf. Krishnamurti, 2003:240). In certain cases, they occur with nouns inflected for case markers. In Telugu and Tamil, it is found that they select similar case markers for certain sets of postpositions whereas it is also evident that they select different case markers for their association with certain other sets. When the case markers selected by postpositions differ between Telugu and Tamil, they cannot be substituted for each other.

Postpositions in languages may have different functions and it is obvious that they lead to mismatches in Machine Translation between a pair of languages. We propose a device, a balance scale for postpositions explicating a functional cline in order to disambiguate the role of them in a particular context and substitute them with appropriate postpositions in the target language.

A balance scale is developed on the basis of the ordering of substitutable elements distributed across a scale with two polar ends, the left and the right poles with a neutral in the middle. A form in the middle is substitutable in case there is no context that is discovered in order to replace with the most appropriate equivalent. Whereas, the left and right pole elements compete for the context. Substitutability of the elements on the balance scale depends on the ontology of nouns and the class of verbs which are used as tools to disambiguate the postposition in the source language.
In this paper, the postpositions (PSP) with their complements marked for the nominative (NOM) and oblique (OBL) or Genitive (GEN) cases are discussed with examples.

3. **Te. NN-NOM PSP => Ta. NN-NOM PSP**

Postpositions in the table (1) occur after the nominative case-marked complements in Telugu and Tamil. These elements are inflected forms of verbs and lost their verbal sense when they occur as postpositions. However, still they retain their verbal meaning when they do not function as postpositions. To distinguish between the syntactic role of a verb and a postposition, the balance scale test is performed.

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Telugu NN-NOM</th>
<th>Tamil NN-NOM</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>lēkuMḍā</td>
<td>illāmal/ ṯṅṛ</td>
<td>‘without’</td>
</tr>
<tr>
<td>2.</td>
<td>kākuMḍā</td>
<td>allāmal/ṛṇṛ</td>
<td>‘except’</td>
</tr>
</tbody>
</table>

**Table 1: Te. NN-NOM PSP and Ta. NN-NOM PSP**

3.1. **Te. lēkuMḍā and Ta. illāmal/ṛṇṛ ‘without’**.

In Tamil, the verb il `be not' occurs with its negative verbal participle form illāmal as a postposition with negative instrumental and negative comitative function (Lehmann, 1993:121). Similarly, the form lēkuMḍā, the verbal participle form of the verb lē functions as a postposition with the above functions in Telugu.

In Tamil, the verb il `be not' occurs with its negative verbal participle form illāmal as a postposition with negative instrumental and negative comitative function (Lehmann, 1993:121). Similarly, the form lēkuMḍā, the verbal participle form of the verb lē functions as a postposition with the above functions in Telugu.

<table>
<thead>
<tr>
<th>1.</th>
<th>Te. nēnu</th>
<th>tālaMcevi</th>
<th>lēkuMḍā</th>
<th>talupu</th>
<th>tīs-</th>
<th>ā-</th>
<th>nu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.NO</td>
<td>key.NO M</td>
<td>without</td>
<td>door.AC C</td>
<td>open</td>
<td>PST-</td>
<td>1.SG</td>
<td></td>
</tr>
<tr>
<td>I.NO</td>
<td>key.NO M</td>
<td>without</td>
<td>door-ACC</td>
<td>open-</td>
<td>PST-</td>
<td>1.SG</td>
<td></td>
</tr>
</tbody>
</table>

'I opened the door without a key.'

<table>
<thead>
<tr>
<th>2.</th>
<th>Te. nēnu</th>
<th>lēkuMḍā</th>
<th>vādu</th>
<th>sinimā-ku</th>
<th>veḷḷ-</th>
<th>ā-</th>
<th>du.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.NOM</td>
<td>without</td>
<td>he.NOM</td>
<td>cinema-DAT</td>
<td>go-</td>
<td>PST-</td>
<td>3.SG.M.</td>
<td></td>
</tr>
</tbody>
</table>
The balance scale test is executed to identify the role of postpositions for the above forms.

**Balance scale test:** When lēkuMḍā in Telugu and illāmal in Tamil occur as a verb, they are replaceable with uMḍakuMḍā (the negative verbal participle form of the verb uMḍu `be') and irukkāmal (the negative verbal participle form of the verb iru `be') respectively. For instance,

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ta.</td>
<td>he.NOM</td>
<td>home-LOC</td>
<td>[be.NEG-CPM/ be-NEG.CPM]</td>
<td>outside</td>
<td>stay-PST-3.SG.M.</td>
<td></td>
</tr>
<tr>
<td>Ta.</td>
<td>avan</td>
<td>vīṭ-il</td>
<td>[il.l-āmal/ iru.kk-āmal]</td>
<td>veliyē</td>
<td>tanik- in-ān</td>
<td></td>
</tr>
<tr>
<td>Ta.</td>
<td>he.NOM</td>
<td>home-LOC</td>
<td>[be.NEG-CPM/ be-NEG.CPM]</td>
<td>outside</td>
<td>stay-PST-3.SG.M.</td>
<td></td>
</tr>
</tbody>
</table>

`He stayed outside having not being at home.'

When lēkuMḍā in Telugu and illāmal in Tamil occur as in postpositions, they cannot be substitutable with uMḍakuMḍā and irukkāmal respectively as in (4).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>key</td>
<td>[witho ut/ be- NEG.CPM]</td>
<td>door.A CC</td>
<td>ope n- PS T- 1.S G.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T a.</td>
<td>nān</td>
<td>cāvi</td>
<td>[illāmal/ *iru.k k- āmal]</td>
<td>katav- ai.t</td>
<td>tīra- nt- ēn.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>key</td>
<td>[witho ut/ be- NEG.CPM]</td>
<td>door- AC C</td>
<td>ope n- PS T- 1.S G.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`I opened the door without a key.'

**3.2. Te. kākuMḍā and Ta. allāmal/aṇṛi `except'.**

The verb al `be not' occurs with its negative verbal participle form allāmal as a postposition, expressing exception `except' (Lehmann, 1993:122). In Telugu, kākuMḍā, the negative participle form of the verb avvu `become' as a postposition expresses similar function of exception.
| 5.  | Te. | nēnu | kākuMḍā | aMdarū | sinimā- | ku | veļ|- | ā- | ru. |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I.NOM | except | everyone | cinema- | DAT | go- | PST- | 3.PL.H. |
| Ta. | nāṇ | allāmal | aṉaiyāvarum | paṭattu- | kku | pō- | Ṇ- | ārkal. |
| I.NOM | except | everyone | cinema- | DAT | go- | PST- | 3.PL.H. |

'Except me, everyone went to the cinema.'

**Balance scale test.**
However, when *kākuMḍā* in Telugu is used as a verb, it can be replaced by *avvakuMḍā* (the negative participle form of the verb avvu `become').

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>egg.NOM</td>
<td>Young one</td>
<td>[become.NEG-CPM/become-NEG.CPM]</td>
<td>escape from-NEG.FUT-3.SG.N.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

'The egg cannot escape form hatching.'

**Balance scale for kākuMḍā as a verb in Telugu:**

avv-akuMḍā >> kā- kuMḍā << *tappa
`become- NEG.CPM >> become.NEG- CPM << `except'

However, when *kākuMḍā* in Telugu and *allāmal* in Tamil occur as postpositions, they alternate with postpositions *tappa* and *tavira* `except' in the respective languages.

| 7.  | Te. | nēnu | [kākuMḍ ā/] | tappa | aMdarū | sinimā- | ku | veļ|- | ā- | ru. |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I.NOM | except | [kākuMḍ ā/] | tappa ] | aMdarū | sinimā- | ku | veļ|- | ā- | ru. |
| [I.NO M | except/ | I- | ACC | except] | everyone | cinema- | DAT | go- | PST-3.PL. H. |

76
"Except me, everyone went to the cinema."

**Balance scale for kākuMḍā as a postposition in Telugu:**
*avv-akuMḍā >> kā- kuMḍā << tappa
`become- NEG.CPM >> become.NEG- CPM << 'except'

**4. NN-OBL PSP => Ta. NN-OBL/GEN PSP**
Certain postpositions require their complements to be marked for the oblique form in Telugu and Tamil. Alternatively, the oblique form of a noun is substituted with the genitive form in Tamil which is not allowed in Telugu.

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Telugu</th>
<th>Tamil</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>guMḍā</td>
<td>valiyāka</td>
<td>`through (a place)'</td>
</tr>
<tr>
<td>2.</td>
<td>dvārā / mūlāna</td>
<td>mūlam</td>
<td>`through (an agent)'</td>
</tr>
<tr>
<td>3.</td>
<td>paṭla</td>
<td>mēl</td>
<td>`with regard to, about, concerning, towards'</td>
</tr>
<tr>
<td>4.</td>
<td>vaipu/vaMka</td>
<td>pakkam</td>
<td>`towards, in the direction of'</td>
</tr>
<tr>
<td>5.</td>
<td>kūḍa</td>
<td>kūṭa</td>
<td>`along with'</td>
</tr>
<tr>
<td>6.</td>
<td>prakāraM</td>
<td>paṭi</td>
<td>`according to'</td>
</tr>
<tr>
<td>7.</td>
<td>veMbaḍi/</td>
<td>u=ṭan/ ō=tu/</td>
<td>`along with, behind, through, from'</td>
</tr>
<tr>
<td></td>
<td>veMṭa</td>
<td>pinnāl/ -il iruntu</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>paṭṭuna</td>
<td>-il/ -ku</td>
<td>`at, in, at the appropriate time'</td>
</tr>
<tr>
<td>9.</td>
<td>mēraku</td>
<td>-kkū/ paṭi</td>
<td>`upto, in accordance with'</td>
</tr>
</tbody>
</table>

Table 2: NN -OBL PSP => Ta. NN-OBL/GEN PSP

4.1. **Te. guMḍā and Ta. vaḷiyāka 'through (a place)'.**
A noun[+place] occurs with the postposition guMḍā in Telugu and vaḷiyāka in Tamil with a verb[+motion] to express through the place.
The postposition valiyāka in Tamil is a grammaticalized form of vai `way' + āka `ADV'. When the noun vai occurs as a complement of the postposition valiyāka, it is optionally dropped. In the case of Telugu, dāri `way' and guMdā `through' can co-occur, since the postposition is not derived from the former one.

```
9. Te.  kukka nā iMți dări guMdā parigett - iM- di.
    dog.NO M  m y house.OB L  way throug h run- PST- 3.SG. N.
Ta.  nāy en vițțu valuți k a  oți y- atu
    dog.NO M  m y house.OB L  throug h run- PST- 3.SG. N.
```

```
`The dog ran through the way of my house.'
```

**Balance scale for noun+[+place]-guMdā in Telugu:**

mārgaMgā >> -lō << *dvārā < *mūlaMgā
`through the passage' >> `LOC' << `through' < `by means of'

**Balance scale for noun+[+place]-valiyāka in Tamil:**

vāyilāka >> -il << *mūlam
`through' >> `LOC' << `by means of'

Nouns+[+container, +passage] with the above postpositions occurring with verbs+[+motion] express the sense of `movement towards and away from' the container.
The balance scale for the noun [+container, +passage] varies from the noun [+place] as where both the polar ends including the middle form of balance scale may replace the given postposition as exemplified below:

**Balance scale for noun [+container, +passage] - guMḍā in Telugu:**

mārgaMgā >> -lō << dvārā < mūlaMgā
'through the passage' >> `LOC' << `through' < `by means of'

**Balance scale for noun [+container, +passage] - vālīyāka in Tamil:**

vāyilāka >> -il << mūlam
`through' >> `LOC' << `by means of'

A noun [+direction] is combined with guMḍā in Telugu to express the sense of `through the noun [+direction]'. Whereas in Tamil, the form -āṟppōl is used in such constructions.

<table>
<thead>
<tr>
<th></th>
<th>Tel.</th>
<th>Te.</th>
<th>ku</th>
<th>eduru</th>
<th>guMḍā</th>
<th>mūrti</th>
<th>vacc</th>
<th>ā</th>
<th>-du.</th>
<th>Ta.</th>
<th>Te.</th>
<th>en-</th>
<th>akku</th>
<th>etir. tt- āṟppōl</th>
<th>mūrtti</th>
<th>va-</th>
<th>nt-</th>
<th>āṟ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-</td>
<td>DAT</td>
<td>In front of</td>
<td>Murti</td>
<td>come-</td>
<td>PST-</td>
<td>3.SG.M.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I-</td>
<td>DAT</td>
<td>In front of</td>
<td>Murti</td>
<td>come-</td>
<td>PST-</td>
<td>3.SG.M.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`Murti came in front of me.'

The postposition guMḍā in Telugu with a noun [+direction] cannot balance the right pole elements of the following balance scale. This proves that it has a different sense comparing to earlier functions. Similarly, the form -āṟppōl in Tamil can balance with the correspondings of Telugu left pole elements.

**Balance scale for noun [+direction] - guMḍā in Telugu:**

-gā > nuMḍi > vaipu > -lō/-na << *mārgaMgā < *dvārā < *mūlaMgā
`ADV' > `ABL' > `towards' >> `LOC' << `through the passage' < `through' < `by means of'

**Balance scale for noun [+direction] - āṟppōl in Tamil:**

-āka > -il iruntu> nōkki > -il/-ē << *vālīyāka < *vāyilāka < *mūlam
`ADV' > `ABL' > `towards' >> `LOC' << `through' < `through' < `by means of'
4.2. Te. dvārā and Ta. mūlam `through (an agent)'.
A noun[+instrument] occurs with the postposition dvārā in Telugu and mūlam in Tamil to express through the agent.

<table>
<thead>
<tr>
<th>12.</th>
<th>Te.</th>
<th>nēnu</th>
<th>tālaMcevi-</th>
<th>dvārā</th>
<th>talupu</th>
<th>teric-</th>
<th>ā-</th>
<th>nu.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I.NOM</td>
<td>key</td>
<td>with</td>
<td>door-</td>
<td>open-</td>
<td>PST-</td>
<td>1.SG.</td>
<td></td>
</tr>
<tr>
<td>Ta.</td>
<td>nāṇ</td>
<td>cāvi</td>
<td>mūlam</td>
<td>katav- ai.t</td>
<td>tira-</td>
<td>nt-</td>
<td>ēn.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I.NOM</td>
<td>key</td>
<td>with</td>
<td>door-</td>
<td>ACC</td>
<td>open-</td>
<td>PST-</td>
<td>1.SG.</td>
</tr>
</tbody>
</table>

'`I opened the door with a key.'`

Balance scale for noun[+instrument]-dvārā in Telugu:

*guMḍā > *mārgaMgā >> -tō << mūlaMgā
`through' > `through the passage' >> `INS' << `by means of'

Balance scale for noun[+instrument]-mūlam in Tamil:

*vaḻiyāka > *vāyilāka >> -āl << mūlamāka
`through' > `through the passage' >> `INS' << `by means of'

Postposition dvārā in Telugu and mūlam in Tamil with a complement noun[+container, +passage] occurring with a verb[+motion] elicits the function of through (the passage). They balance with postpositions guMḍā and vaḻiyāka `through' in Telugu and Tamil respectively.

4. 3. Te. paṭla and Ta. mēl `with regard to, about, concerning, towards'.

Postposition paṭla in Telugu and mēl (lit. `on') in Tamil refers to meanings such as `with regard to', `about', `concerning' and `towards'. They express adnominal relationship with the noun[+emotional state] in a sentence.

They occur adnominally preceding the subcategorized argument noun[+emotional state] of a transitive verb. As in the example (13) , they modify the head noun of the following phrase i.e. āsakti and ārvam `interest' in Telugu and Tamil respectively.
The teacher noticed her interest [with regard to/about/concern/towards] studies.

With the dative-marked subject construction, a noun marked for paṭla in Telugu and mēl in Tamil has an adnominal relationship with the noun [+emotional state] which agrees with the verb as in the example (14).

<table>
<thead>
<tr>
<th>Te.</th>
<th>mūrti-ki</th>
<th>ḍabbu</th>
<th>paṭla</th>
<th>āśa</th>
<th>uMdi.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murti</td>
<td>DAT money</td>
<td>towards</td>
<td>desire</td>
<td>have</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ta.</th>
<th>mūrti-kku</th>
<th>paṇatt-iṇ</th>
<th>mēl</th>
<th>ācai</th>
<th>ullatu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murti</td>
<td>DAT money-GEN</td>
<td>on</td>
<td>desire</td>
<td>have</td>
<td></td>
</tr>
</tbody>
</table>

`Murti has a desire towards money.'

With verbs of intransitive, nouns marked with paṭla in Telugu and mēl in Tamil have an adnominal relationship with the noun [+emotional state] inflected for the adverbializer or associative marker as in example (15).

<table>
<thead>
<tr>
<th>Te.</th>
<th>mūrti-nā-</th>
<th>paṭla</th>
<th>kōpaM</th>
<th>[gā/tō]</th>
<th>unn-</th>
<th>ā-</th>
<th>ḍu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murti</td>
<td>I.OB L</td>
<td>toward s</td>
<td>angry-</td>
<td>[ADV/ASS]</td>
<td>be-</td>
<td>PST</td>
<td>3.SG.M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Murti</td>
<td>I.OB L</td>
<td>on</td>
<td>angry-</td>
<td>[ADV/ASS]</td>
<td>be-</td>
<td>PST</td>
<td>3.SG.M</td>
<td></td>
</tr>
</tbody>
</table>

`Murti is angry towards me.'

**Balance scale for subject-NOM/DAT + noun-OBL-paṭla + noun [-abstract, +emotional state] in Telugu:**

pai > mīda >> yaMdu << *tō < *valla
`on' > `on' >> `in' << `INS' < `because of'

Balance scale for subject-NOM/DAT + noun-OBL/GEN-mēl + object [-abstract, +emotional state] in Tamil:

mēl > mītu >> -il/-iṭam << *āl
`on' > `on' >> `LOC' << `INS'

4. 5. Te. vaMka/ vaipu and Ta. pakkam `towards, in the direction of'.
Postpositions vaMka/ vaipu in Telugu and pakkam in Tamil with the verb[+perception] express `towards or in the direction of'.

<table>
<thead>
<tr>
<th>16.</th>
<th>Te.</th>
<th>mūrti</th>
<th>rāDa</th>
<th>vaMka</th>
<th>cūs-</th>
<th>ă-</th>
<th>ɗu.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Murti</td>
<td>Radha.OBL</td>
<td>towards</td>
<td>see-</td>
<td>PST-</td>
<td>3.SG.M.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ta.</td>
<td>mūrtti</td>
<td>rātā.(v</td>
<td>-iṇ)</td>
<td>pakkam</td>
<td>pār-</td>
<td>tt-</td>
</tr>
<tr>
<td></td>
<td>Murti</td>
<td>Radha.OBL-</td>
<td>(GEN)</td>
<td>towards</td>
<td>see-</td>
<td>PST-</td>
<td>3.SG.M.</td>
</tr>
<tr>
<td></td>
<td>`Murti saw towards Radha.'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Te. kūḍa and Ta. kūṭa `along with'.
Postpositions kūḍa in Telugu and kūṭa in Tamil are marked with the oblique form of a noun to express the comitative function when occurring with a verb[+motion].

<table>
<thead>
<tr>
<th>17.</th>
<th>Te.</th>
<th>mūrti</th>
<th>nā-</th>
<th>kūḍa</th>
<th>iMṭi-</th>
<th>ki</th>
<th>vacc-</th>
<th>ă-</th>
<th>ɗu.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Murti.NOM</td>
<td>I.OBL-</td>
<td>ASS</td>
<td>house-</td>
<td>DAT</td>
<td>come-</td>
<td>PST-</td>
<td>3.SG.M.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ta.</td>
<td>mūrtti</td>
<td>en-</td>
<td>kūṭa</td>
<td>viṭṭ-</td>
<td>ukku</td>
<td>va-</td>
<td>nt-</td>
<td>āṇ.</td>
</tr>
<tr>
<td></td>
<td>Murti.NOM</td>
<td>I.OBL-</td>
<td>ASS</td>
<td>house-</td>
<td>DAT</td>
<td>come-</td>
<td>PST-</td>
<td>3.SG.M.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`Murti came along with me to the house.'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. Te. prakāraM and Ta. paṭi `according to'
Nouns prakāraM in Telugu and paṭi in Tamil occur as postpositions to express `according to'.

<table>
<thead>
<tr>
<th>18.</th>
<th>Te.</th>
<th>nāṇna</th>
<th>īṣṭa</th>
<th>prakāraM</th>
<th>peḷḷi</th>
<th>jarig-</th>
<th>iM-</th>
<th>di.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>father</td>
<td>wish.OBL</td>
<td>accordin</td>
<td>marriag</td>
<td>happen-</td>
<td>PST-</td>
<td>3.SG.N.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ta.</td>
<td>appā.</td>
<td>īṇ</td>
<td>viruppa(tt-</td>
<td>īṇ)</td>
<td>paṭi</td>
<td>kalyāṇam</td>
<td>naṭa-nt- atu.</td>
</tr>
</tbody>
</table>
According to the desire of my father, the marriage has taken place.'

7. Te. veMbaḍi/ veMṭa and Ta. u\=taṇ/ Ṽṭu/ piṅṇāl/ -il iruntu `along with, behind, through, from'

The postposition veMbaḍi/ veMṭa in Telugu has distinct functions. In Tamil, there is no direct equivalent for the corresponding Telugu one. Here, the balance scale of Telugu is used as tool to disambiguate it in order to get the equivalent Tamil postposition.

In Telugu, nouns [+animate] is marked for the postposition veMbaḍi/ veMṭa with verbs [+motion, -run] to express the comitative function `along with'. Whereas in Tamil, the associative case marker is marked with nouns [+animate].

Balance scale for nouns [+animate]-veMbaḍi/ veMṭa+verbs [+motion, -run] in Telugu:

tōpāṭu > kūḍa >> veMbaḍi/veMṭa << *venuka
`ASS' > `along' >> `along' << `behind'

Nouns [+animate] is marked for the postposition veMbaḍi/ veMṭa with verbs [+motion, +run] to express the location `behind'. In this case, the postposition piṅṇāl occurs with nouns in Tamil.

<table>
<thead>
<tr>
<th>father</th>
<th>GEN wish.</th>
<th>(GEN) accordi ng to</th>
<th>marriage.N OM</th>
<th>happe n-PST-3.SG. N.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'According to the desire of my father, the marriage has taken place.'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Te. nā [veMbaḍi/veMṭa]</th>
<th>kumār</th>
<th>vas-</th>
<th>tunnā-</th>
<th>đu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.OBL along with Kumar</td>
<td>come-</td>
<td>DUR-3.SG.M.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ta. eṇ.ŋ-utan</td>
<td>kumār</td>
<td>varu-</td>
<td>kir-</td>
<td>än</td>
</tr>
<tr>
<td>I ASS</td>
<td>Kumar</td>
<td>come-</td>
<td>DUR-3.SG.M.</td>
<td></td>
</tr>
</tbody>
</table>

`Kumar comes along with me.'

<table>
<thead>
<tr>
<th>Te. pōlisu doMga [veMbaḍi/ veMṭa]</th>
<th>parigett-</th>
<th>ā-</th>
<th>đu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>police thief behind run-</td>
<td>PST-3.SG.M.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Balance scale for nouns [+animate] - veMbaḍi/ veMṭa + verbs [+motion, +run] in Telugu:

*tōpātu > *kūḍa >> veMbaḍi/veMṭa << venuka
`ASS' > `along' >> `along' << `behind'

Nouns [+place] is marked for the postposition veMbaḍi/ veMṭa in Telugu to express the direction `along'. The associative case marker ōṭu in Tamil is substituted for this context (Cf. Lehman, 1993:38 example(83)).

Balance scale for nouns [+place] - veMbaḍi/ veMṭa in Telugu:
*tōpātu > *kūḍa >> veMbaḍi/veMṭa << guM\bdada
`ASS' > `along' >> `along' << `through'

Nouns [+body parts] is marked for the postposition veMbaḍi/ veMṭa in Telugu to express the ablative function `from'. The ablative case marker -il iruntu in Tamil can occur in this context.
8. Te. paṭṭuna and Ta. -il, -ku `at, in, at the appropriate time.'

The postposition paṭṭuna is marked on a noun[+place] to denote the location at or in. In Tamil, the locative marker is substituted for the corresponding Telugu one, since there is no postposition to equate it.

<table>
<thead>
<tr>
<th>23.</th>
<th>Te.</th>
<th>vādu</th>
<th>iMṭi</th>
<th>paṭṭuna</th>
<th>unn-</th>
<th>ā-</th>
<th>ṭu.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>he.NOM</td>
<td>home.OBL</td>
<td>at</td>
<td>be-</td>
<td>PST-</td>
<td>3.SG.M</td>
<td></td>
</tr>
<tr>
<td>Te.</td>
<td>avaṇ</td>
<td>viṭṭ-</td>
<td>il</td>
<td>iru-</td>
<td>kkiṟ-</td>
<td>āṇ.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>he.NOM</td>
<td>home.OBL</td>
<td>LOC</td>
<td>be-</td>
<td>PST-</td>
<td>3.SG.M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`He is at home.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When paṭṭuna governs a complement noun[+time], it expresses the sense of `at the appropriate time'. The dative marker in Tamil expresses the similar function.

<table>
<thead>
<tr>
<th>24.</th>
<th>Te.</th>
<th>nuvvu</th>
<th>vēla</th>
<th>paṭṭuna</th>
<th>tin-</th>
<th>āli.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>you.NOM</td>
<td>time.OBL</td>
<td>in</td>
<td>eat-</td>
<td>FORC.</td>
<td></td>
</tr>
<tr>
<td>Ta.</td>
<td>nī</td>
<td>nērattu-</td>
<td>kku</td>
<td>cáppiṭ-</td>
<td>a-</td>
<td>vēntum.</td>
</tr>
<tr>
<td></td>
<td>you.NOM</td>
<td>time-</td>
<td>DAT</td>
<td>eat-</td>
<td>INF-</td>
<td>FORC.</td>
</tr>
<tr>
<td></td>
<td>`You have to eat on time.'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. Te. mēraku and Ta. -kku/ paṭi `upto, in accordance with'

The postposition mēra(ku) in Telugu expresses the limitation of time, when it occurs with a noun[+time]. In Tamil, the dative is marked on noun[+time] to express the similar function.

<table>
<thead>
<tr>
<th>25.</th>
<th>Te.</th>
<th>nēnu</th>
<th>reMḍu</th>
<th>ēḷḷa</th>
<th>mēra(ku)</th>
<th>selavu</th>
<th>peṭṭ-</th>
<th>ā-</th>
<th>nu.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I.NO M</td>
<td>two years.OBL</td>
<td>upto</td>
<td>leave</td>
<td>keep</td>
<td>PST-</td>
<td>1.SG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ta.</td>
<td>nāṇ</td>
<td>iranṭu</td>
<td>varuṭaṅkaḷ</td>
<td>uku</td>
<td>viṭumura</td>
<td>ēṭu-</td>
<td>tt-</td>
<td>ēṇ.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I.NO M</td>
<td>two years-</td>
<td>DAT</td>
<td>leave</td>
<td>take-</td>
<td>PST-</td>
<td>1.SG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>`I took leave for two years.'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Postpositions mēra(ku) in Telugu and paṭi in Tamil express the function of `in accordance with'.
5. Conclusion
Similar to case markers, it is found that postpositions too exhibit different functions. To disambiguate these postpositions, a balance scale is developed which substitutes the given postposition in a functional cline with two polar ends. In certain context the left polar words can be substituted to the given postposition, whereas in another context, the right polar words can be substituted. A form in the middle is substitutable in case there is no context discovered. In this way, postpositions from Telugu to Tamil is substituted to obtain better results in the current MT.

References


Complexities in Developing Tamil-Brahmi Script OCR: An Analysis
M Monisha¹, V S Felix Enigo²
¹Research Scholar, ²Associate Professor
¹,²Department of Computer Science and Engineering
¹,²SSN College of Engineering, Chennai
¹moni.munivel@outlook.com, ²felixvs@ssn.edu.in

Abstract
Brahmi script is one of the oldest scripts in the world which is written in Tamil, Prakrit and Pali languages. Tamil is the oldest language among the three which is rich in literature and heritage. Many historic pieces of evidences which was written in Tamil on various medium such as potsherds, stone, palm leaves, rock, etc. have been ruined due to nature’s disasters and negligence. But recently, due to various governmental efforts, Brahmi script epigraphs have been documented as inscriptions images and was preserved. These epigraphs need epigraphists to understand the text. But currently very few epigraphist are present to decode the text meaningfully to modern Tamil language. Hence, there is a need for an alternate automated way of recognizing these inscriptional Brahmi script to know about the ancient culture for the present and future generations. Optical Character Recognition (OCR) is one such automated technique that extracts the text in the inscription image to recognizable characters. In this paper, various problems that may arise during developing a OCR for Tamil-Brahmi script have been extensively analysed and discussed.

Keywords: Tamil-Brahmi Script, Palaeography, Orthography, OCR, Epigraphs, Potsherds, Inscriptions

1. Introduction
In ancient times informations are written as inscriptions in rock pillars, temple walls, copper plates, pottery, palm leaves, stones, etc. These inscriptions give us valuable information about the cultural, religious actions, socio-economic conditions of the people and the administrative and the political skills of various dynasties in ancient days. But, the Tamil-Brahmi script being the oldest script was found only in potsherds discovered from excavation, surface exploration and
few as cave inscriptions [9]. These informations were preserved as an inscription images at various centres of Archaeological Survey of India. At present a very few epigraphist were found who can decode and understand these script. Hence, an automatic character recognition of Brahmi script becomes essential to know these epigraphical information.

Optical character recognition (OCR) is an automated character recognition technique that converts the text in the image into a machine readable text. At present, OCR has been developed in many ancient and modern international languages. But, so far OCR was not developed for Brahmi script due to various complexities in the script. Many problems come up while developing an Optical Character Recognition tool for Brahmi script. These issues can be categorized as paleographic, orthographic and morphophonemes issues. Paleographic issues are the issues due to variation in structure of the character over different periods and the direction of the writing. Some striking structural issues of Brahmi script are, it does have dots (pulli) and early period Brahmi characters does not have variation for short and long vowel. Yet another orthographic issue is the divider symbol “|” used for sentence divider is often misunderstood for the character “ra”. Morphophonemes are the changes that occur when two morphemes join together. Brahmi script use this concept only for identification of periods not in a grammatical sense as in Tamil grammar [5].

In this paper, the various complexities involved in developing an OCR for Brahmi script is discussed elaborately. This paper is organized as follows: Section II discusses the genesis of Brahmi script and the early Tamil language. Section 3 deals with the characteristics and complexities of Tamil-Brahmi script. Section 4 describes the issues arises in recognizing the Tamil-Brahmi script. Finally, Section 5 concludes the paper with the p for future work.

2. Brahmi Script and Tamil Language

Language is a medium to express one’s view and to communicate with one another. Script is used to develop or write a language. Brahmi script is one of the oldest scripts in the world. In India, Brahmi Script was seen in many ancient inscriptions and antiquity excavated. Brahmi Script is written in three languages: Tamil, Prakrit and Pali. Among this, Tamil is the oldest spoken language in the world which has rich literatures [2] [7].

Tamil-Brahmi script has alphabetical and syllabic systems [4] with relatively small character set of pure 12 vowels and 18 consonants. Additionally, it consists of 6 consonants known as Grantha letters which were borrowed from Prakrit language. Apart from this, a character called Adytam, neither a consonant nor a vowel is used in Tamil Grammar [2] [7]. Generally, when one or more
consonants combines with vowels or two consonants cluster, it leads to a modified shaped character called vowel diacritic [2].

2.1 History of ancient Brahmi Script and Inscriptions

Brahmi script writing systems is mostly seen in early period inscriptions. The earliest inscriptions discovered is the post-Indus corpus which was written in Brahmi script [1] [3]. The rock-cut edicts of Ashoka in North-Central India, dated 250-232 BCE is the best known earliest Brahmi Inscriptions in India. In the 5th century BCE, the Brahmi script evolved as a short Brahmi script. The short Tamil-Brahmi script, which dated 540 BCE is found in Palani, South India. It is also seen in inscriptions found in Sri Lanka, Thailand and Egypt written in the 1st century BCE and 2nd century CE [3]. Region-wise statistics of where the Brahmi inscriptions were found is given in Table 1 and a Brahmi rock inscription is shown in Figure 1.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Region</th>
<th>Name of the Region</th>
<th>Sites</th>
<th>Inscriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Northern Region</td>
<td>Thondai Nadu</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Southern Region</td>
<td>Pandiya Nadu</td>
<td>24</td>
<td>71</td>
</tr>
<tr>
<td>3</td>
<td>Western Region</td>
<td>Cera Nadu</td>
<td>5</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>Eastern Region</td>
<td>Chola Nadu</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1. Brahmi Inscription sites

Fig 1. Alagarmalai Inscription

Table 2 shows the chronological arrangement of Tamil-Brahmi inscriptions based on palaeographic (ancient handwriting), orthographic (conventions for writing language) and linguistic (language science) evidence.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Period</th>
<th>Sites</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Early Period</td>
<td>Aiyarmalai, Alagaramalai, Arittapatti, Karungalakudi, Kilavalavu, Kilkuyilkudi, Kongarpuliyangulam, Mangulam, Marukaltalai, Mettupatti, Mudalaikulam, Muttupatti,</td>
</tr>
</tbody>
</table>
2.2 Evidences of Tamil-Brahmi Script in Pottery and Potsherds

Till the middle of 20th century, the cave inscriptions were the only records for the Tamil-Brahmi script. The first pottery inscriptions was discovered at Arikamedu between 1941 – 1944. Later it was found in many sites across Tamil Nadu, Pondicherry and Kerala. At Present, out of 27 sites found, 19 have been excavated and the remaining were surface explorated. Major 7 sites where Tamil-Brahmi pottery inscription excavated is listed in Table 3. Apart from this, few inscriptions have been excavated from Kanchipuram, Karur, Korkai and Poompuhar. The pottery inscription from Kanchipuram was written in Prakrit and from Poompuhar was written in Pali. The sites Alagarai, Kovalanpottal, Poluvampatti, Vallam, Maligaimedu and Teriruveli has one pottery inscription each. Figure 2 shows Tamil-Brahmi potsherd inscriptions.

<table>
<thead>
<tr>
<th>Sl. No</th>
<th>Site Name</th>
<th>District</th>
<th>Riverbank</th>
<th>Excavation period</th>
<th>No. of Potteries</th>
<th>Dated on palaeographic evidence</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arikamedu</td>
<td>Virampatnam</td>
<td>Ariyankuppam</td>
<td>1941-44, 1947-50, 1989-92</td>
<td>66</td>
<td>2BCE-3CE</td>
<td>Tamil, Prakrit and Pali</td>
</tr>
<tr>
<td>2</td>
<td>Uraiyyur</td>
<td>Tiruchira palli</td>
<td>cauvery</td>
<td>1965-69</td>
<td>20</td>
<td>1 BCE</td>
<td>Tamil</td>
</tr>
<tr>
<td>4</td>
<td>Alagankulam</td>
<td>Rameswaram</td>
<td>vaigai</td>
<td>1986-1987</td>
<td>Graffiti marks</td>
<td>2BCE-1CE</td>
<td>Tamil, Pali</td>
</tr>
</tbody>
</table>

Table 2. Chronological arrangement of Tamil-Brahmi inscriptions
### Table 3. Various sites of Tamil-Brahmi pottery inscription in Tamil Nadu

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Site Name</th>
<th>Country</th>
<th>Potsherds dated</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Salihundam</td>
<td>srikulam</td>
<td>Vamsadhara</td>
<td>1950</td>
</tr>
<tr>
<td>6</td>
<td>Pattanam</td>
<td>ernakulam</td>
<td>Periyar</td>
<td>2010-2011</td>
</tr>
<tr>
<td>7</td>
<td>Keezhadi</td>
<td>Sivaganga</td>
<td>Vaigai</td>
<td>2015-Present</td>
</tr>
</tbody>
</table>

Table 4. Potsherds found in various countries

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Site Name</th>
<th>Country</th>
<th>Potsherds dated</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Poonagari, Jaffna</td>
<td>Sri Lanka</td>
<td>2 BCE</td>
<td>Tamil</td>
</tr>
<tr>
<td>2</td>
<td>Quseir-al-Qadim</td>
<td>Egypt</td>
<td>1 CE</td>
<td>Prakrit</td>
</tr>
<tr>
<td>3</td>
<td>Berenike</td>
<td>Egypt</td>
<td>1 CE</td>
<td>Prakrit</td>
</tr>
<tr>
<td>4</td>
<td>Khor Rori</td>
<td>Oman</td>
<td>1 CE</td>
<td>Tamil</td>
</tr>
<tr>
<td>5</td>
<td>Krabi</td>
<td>Thailand</td>
<td>2 CE</td>
<td>Tamil</td>
</tr>
</tbody>
</table>

3. Characteristics of Tamil-Brahmi Script

The inscriptive Tamil language is very intriguing despite of its complexity. The rise of Tamil as an epigraphic language is seen as a dynamic process which evolved constantly alongside with the literary varieties in terms of palaeographic...
and orthographic (lexical, syntactic and semantic) features [6]. In Tamil-Brahmi script, apart from vowels and consonants, many characters are formed by combining them which are called as Compound characters.

Compound characters are modified characters that are formed, when a vowel and a consonant are combined, depending upon the location of the vowel marker. Compound characters have compound orthographic shape that is formed by altering the basic shape of the character by adding any one of the following to the consonant: a) vowel marker b) vowel-specific suffix or prefix and c) vowel-specific suffix and prefix [7]. Mantras are compound characters where the same consonant combines with different vowels and these characters are written with additional strokes. Two consonants combines to form a consonant cluster called as Nexus [1]. As most of the characters are formed by very little variation in there shape, OCR is challenging in Tamil-Brahmi script resulting in very low recognition [7].

3.1 Problems in Digitizing Tamil-Brahmi Script

Digitizing ancient Tamil-Brahmi script inscription images using Optical Character Recognition is a tedious process. It is due to the inherent complexity of the script and also it is handwritten. The script complexities are: 2D structure, Variation in Stroke, Stroke order variations, Symbol order variations, Number and Direction variations [1] and Inter-class similarity. Complexities induced due to writer are: broken characters, characters touching each other, uneven character intensity, skewed characters, and overlapped characters. Additionally, noise and distortion due to environment and mood of the writer [8] effects the recognition. It complexities are listed in Table 5.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Difficulties</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Similar Shapes by Writers</td>
<td>b(pha), d(va), j(ca)</td>
</tr>
<tr>
<td>2</td>
<td>Unnecessary Strokes</td>
<td>l(u), l(ra), l'(ray)</td>
</tr>
<tr>
<td>3</td>
<td>Minor Variation</td>
<td>f(nga), f(u)</td>
</tr>
<tr>
<td>4</td>
<td>Shape Variation</td>
<td>l(u), l(ua)</td>
</tr>
<tr>
<td>5</td>
<td>Angle Variation</td>
<td>l(u), l(pa)</td>
</tr>
<tr>
<td>6</td>
<td>Character Continuity</td>
<td>f(daa), f(ja)</td>
</tr>
<tr>
<td>7</td>
<td>Similar in Look</td>
<td>i(ni), i(ngi)</td>
</tr>
<tr>
<td>8</td>
<td>Curve Variation</td>
<td>l(pa), b(pha)</td>
</tr>
<tr>
<td>9</td>
<td>Double Characters</td>
<td>m(f)(Shri)</td>
</tr>
</tbody>
</table>

Table 5. Complexities of Brahmi Script due to script and writer

These issues can be of palaeographic, orthographic and morphophonemes.
3.2 Palaeographic Complexities of Tamil-Brahmi Script

Palaeography is the study and process of ancient and historical handwriting to decipher and read historical manuscripts. Palaeographic variations such as voiced consonants, aspirates, sibilants, anusvara and visarga was absent in Tamil-Brahmi script. Some of the characters are modified with the diacritic mark pulli to represent the basic consonants. This diacritic mark is used to distinguish the short vowels ‘e’ and ‘o’ from the respective long vowels and avoid the ligatures between the consonant clusters.

The direction of writing of Brahmi script in inscription is engraved normally from left to right. But, Kilavavu inscription is engraved on the top of the cave. These scribes are engraved from right to left with the letters turned upside down, so that the monks in the cave looking up at the inscriptions could read them normally. In kunnakudi, the inscription is engraved from left to right but the letters are written upside down. Menhir (memorial stone) from Pulimaankombai has three lines written from bottom to top and the direction of reading is from left to right.

While developing Tamil-Brahmi OCR, these paleographic issues should be considered for better recognition accuracy.

3.3 Orthographic Issues in Tamil-Brahmi Script

Orthography is a set of conventions to write a language. It includes spelling normalization, hyphenation, word breaks and punctuation. Based on palaeographic and linguistic considerations, Tamil-Brahmi inscriptions reveals two orthographical systems in two successive periods: earlier orthographical system (3-1 BCE) and later orthographical system (1-6 CE). In early orthographic period, along with the above palaeographic changes, several orthographic modifications in the notation of medial vowels were also seen in Tamil Brahmi script. There is no distinction between short and long ‘e’ and ‘o’ in the Tamil Brahmi script though the difference exists in the language. The consonantal symbol that is unaccompanied by a medial sign represents the basic consonants. The medial sign also denotes the medial vowel.

This proves that pulli must have been used in later period to indicate the basic consonant. pulli is a diacritical mark invented to eliminate the confusion between the basic consonant and the consonant with the inherent ‘a’. The theoretical deduction also confirms that the actual occurrence of pulli is seen at the end of the earlier orthographical system. Also, study of inscriptions reveals that the early Tamil writing was experimented with different orthographic systems for denoting medial vowels, before settling down to the system described in classical Tamil grammar.
Usually, the punctuation marks are not seen Tamil-Brahmi inscriptions. But, a single vertical line which acts as a divider between two sentences is used in inscription at Tirupparankunram. But taking into account the periods and context, these issues should be resolved during the development of Tamil-Brahmi OCR.

Brahmi script is a syllabary in which each akshara is an open syllable which is either a vowel or ends with a vowel, except in the case of the anusvara. If the basic consonant is muted, it is not considered as an akshara and cannot stand by itself. These principles are consolidated into a writing system with the following set of conventions:

1. A consonantal symbol is invested with the inherent –a
2. The notational system of medial vowels commences only with the medial –a, since –a is inherent and does not require a marker
3. A basic consonant cannot be represented except as part of a conjunct consonant
4. A conjunct consonant being an open syllable cannot depict a consonant in the final position

4. Problems in Recognising Brahmi Script
4.1 Doubling the Consonants Issues

Tamil-Brahmi script does not have dotted consonants (mei eluthu) as in modern Tamil language. Hence, during OCR, places where dotted consonant is missed in the word, the consonant of the word is doubled and inserted as a dotted consonant prefix to the consonant. For Example, in ‘upu’ the consonant ‘p’ is doubled and added to u(p)pu. The rules for doubling the consonants for Tamil-Brahmi script is given in Table 6.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Rule</th>
<th>Description</th>
<th>Inscription Text</th>
<th>Anticipated Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Consonant in Tamil Brahmi</td>
<td>Consonants doubled</td>
<td>Koṭupitōn ceypiṭa upu tacaṇ</td>
<td>koṭṭupi(t)tōn ceypi(t)ta u(p)pu ta(c)cana</td>
</tr>
<tr>
<td>2</td>
<td>Consonants in the loanwords</td>
<td>Consonants doubled in loanwords (words borrowed from Prakrit)</td>
<td>upa(can dhamam puta atittanam</td>
<td>upa(c)can dha(m)mam pu(t)ta atittan(n)am</td>
</tr>
<tr>
<td>3</td>
<td>Consonants as a grammatical</td>
<td>Consonants doubled as per the</td>
<td>Amanan katumi</td>
<td>ama(n)nan katu(m)mi</td>
</tr>
</tbody>
</table>
Doubling of the final consonant
The final consonant of a monosyllabic stem is doubled when followed by a vowel

Doubling of the initial consonant
The initial consonant of the succeeding part is doubled with a consonant

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Analytical Writing</th>
<th>Types</th>
<th>Inscription text</th>
<th>Anticipated Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A consonant followed by a vowel</td>
<td>Between word segments</td>
<td>Kutal-ur</td>
<td>Kutalur</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between word and suffix</td>
<td>Katal-an</td>
<td>Katalan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between word segment / suffix</td>
<td>Ay-am</td>
<td>ayam perayam</td>
</tr>
<tr>
<td>2</td>
<td>Vowel followed by a vowel</td>
<td>Between word-segments</td>
<td>Karu-ur</td>
<td>karu(v)ur</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between word and suffix</td>
<td>Ati-on</td>
<td>ati(y)on</td>
</tr>
</tbody>
</table>

Table 7. Rules based on Analysis of writings in Tamil-Brahmi inscriptions

4.3 Orthographic changes in loanwords
Tamil-Brahmi script borrowed many loanwords from the Prakrit language. Such words are not present in modern tamil language. Hence, when performing OCR from Tamil-Brahmi script to Tamil, these words should be replaced by an equivalent tamil words with similar voice as shown in Table 8.
<table>
<thead>
<tr>
<th></th>
<th>Anticipated Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Loss of voicing Caused by a voiceless contiguous consonants g → k d → t b → p</td>
</tr>
<tr>
<td></td>
<td>ganaka → (k)anaka danam → (t)ana kutumbika → kutum(p)ikan</td>
</tr>
<tr>
<td>2</td>
<td>Loss of aspiration Lost at the beginning of a stressed syllable when the voiceless plosives gh→k jh→c th→t</td>
</tr>
<tr>
<td></td>
<td>ghatika → (k)atika upajha → upa(c)an adhithana → ati(tt)anam</td>
</tr>
<tr>
<td>3</td>
<td>Loss of anusvara (m) Characterizes sound produced by lowering the soft palate</td>
</tr>
<tr>
<td></td>
<td>End with M Danam → Tana</td>
</tr>
<tr>
<td>4</td>
<td>Loss of h Deletion of voiceless glottal fricative or /h/ sound a→ha</td>
</tr>
<tr>
<td></td>
<td>ariti → hariti aritan → haritan</td>
</tr>
</tbody>
</table>

Table 8. Orthographic changes in loanwords

4.4 Issues in Tamil-Brahmi Morphophonemics

Morphophonics is the study of changes that occur in the process of joining morphemes in a word or words in a sentence. Morphemes in a word is called as internal sandhi and between words is called as external sandhi. Internal sandhi is seen in early period inscriptions and external sandhi in the later period inscriptions. Different ways the sandhi can change is given below:

1. **Sandhi without changes**: u(p)pu vanikan
2. **Sandhi of consonant and vowel**: elam+per+atan and ur+atai
3. **Mutation of the initial stem in sandhi**: peru → per+atan
4. **Changes in Sandhi**: The sandhi changes based on final sound of the preceding part and the initial sound of the succeeding part. Sandhi may occur as Assimilation (tirital), Addition (tonral) and Elision (ketutal).

4.4.1 Assimilation

Assimilation is the sound change in which some phonemes change is more similar to nearby sounds. It occurs either within words and between words. There are two types of assimilation namely, regressive and progressive assimilation. Regressive assimilation is the changes made in following word segment. Whereas, progressive assimilation is the changes made with respect to the preceding word segment. But, Tamil Brahmi inscriptions has regressive and mutual assimilation, but not progressive assimilation. Mutual assimilation is a
hybrid assimilation where the changes occur in both preceding and succeeding consonants of the word segments. Since, such rules are not available in modern Tamil language. OCR recognizes the word segments without any change as in the Tamil-Brahmi script. For Example, ilam + kāyipān read as ilam kāyipān not as ilaṅkāyipān. Table 9 shows the effect of assimilation in Tamil-Brahmi script.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Assimilation</th>
<th>Rules</th>
<th>Inscription Text</th>
<th>Anticipated Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Regressive assimilation</td>
<td>m+k → ŋk</td>
<td>ilam + kāyipān perum + katu + kōn</td>
<td>ilaṅkāyipān peruṅkatuṅkōn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m+c → ŋc</td>
<td>ilam+catikān netum+caliyān</td>
<td>ilaṅcatikān netuṅcaliyān</td>
</tr>
<tr>
<td></td>
<td></td>
<td>l+k → ŋk</td>
<td>vel + kasipān vel + kācipān</td>
<td>venkāsipān venkācipān</td>
</tr>
<tr>
<td></td>
<td></td>
<td>l+p → np</td>
<td>vel + palli</td>
<td>Venpalli</td>
</tr>
<tr>
<td></td>
<td></td>
<td>l+n → nn</td>
<td>el + nai(ney)</td>
<td>Ennai</td>
</tr>
<tr>
<td>2</td>
<td>Mutual assimilation</td>
<td>l + tt → rr</td>
<td>vempil + tt + ur</td>
<td>vempirr-ur</td>
</tr>
</tbody>
</table>

Table 9. Assimilation in Tamil-Brahmi Script

4.4.2 Addition of Glides and Sandhi

The addition is the insertion of glides between two vowels when the preceding one is a front vowel or a back vowel. Glide –y is added to the noun ending with –i or –ai and glide –v- is inserted between two vowels when the preceding vowel is the back vowel. Table 10, shows the rules for addition of glides.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Addition</th>
<th>Rule</th>
<th>Inscription Text</th>
<th>Anticipated Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Addition of glide</td>
<td>Glide –y</td>
<td>karui+a elai+ur</td>
<td>Karuiya elaiyur</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Palli + ai</td>
<td>palliyaì</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Glide –v</td>
<td>illa + on kuna + in iru + ar</td>
<td>illavon kunavin iruvar</td>
</tr>
<tr>
<td>2</td>
<td>Addition in sandhi</td>
<td>–m → -n</td>
<td>pa+m+kat+a</td>
<td>paṅka(t)a</td>
</tr>
</tbody>
</table>

Table 10. Addition of Glides and Sandhi
4.4.3 Elision

Elision is the omission of one or more sounds in the word or phrase. It is also called as deletion in the historical languages. Elision usually occurs at the stem final character as shown in Table 11. Elision is not present in Tamil-Brahmi script. So, after OCR the words in the script is recognized as two words without change instead of single combined word.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Elision Rule</th>
<th>Inscription Text</th>
<th>Anticipated Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>stem final vowel before a succeeding vowel</td>
<td>kana + tikan</td>
<td>kana tikan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tonṭi + llavō</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>stem final suffix (a)m followed by the noun</td>
<td>Kalam + nel pāṇitam + vānīkan</td>
<td>kalanel pāṇitavānīkan</td>
</tr>
<tr>
<td>3</td>
<td>stem final suffix (a)n followed by the noun</td>
<td>arattan + kāyipana kanakana + atan</td>
<td>arattkāyipana kanakatan</td>
</tr>
<tr>
<td>4</td>
<td>stem final suffix (a)n before a genitive suffix</td>
<td>Cēntan + ā Kuviran + ā</td>
<td>Cēntana Kuvirāa</td>
</tr>
</tbody>
</table>

Table 11. Elision of a stem character

4.5 Other Grammar Issues

Other common grammar issues in Tamil-Brahmi script that should be considered while performing OCR are tabulated in Table 12.

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Rule</th>
<th>Inscription Text</th>
<th>Anticipated Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Disappearance of Ikaram</td>
<td>makaṇ +</td>
<td>anaṅkatuṅkō</td>
</tr>
<tr>
<td></td>
<td></td>
<td>katuṅkō +</td>
<td>anaṅkatuṅkō</td>
</tr>
<tr>
<td>2</td>
<td>Disappearance of An suffix</td>
<td>kana + atan</td>
<td>kuvira antai</td>
</tr>
<tr>
<td>3</td>
<td>Sprinkling of ikara yakara</td>
<td>palī'iy</td>
<td>karu'iyā</td>
</tr>
<tr>
<td></td>
<td>(Ikara yakara ṭutu viravutal)</td>
<td>karu'iyā</td>
<td>valuttiy</td>
</tr>
<tr>
<td>4</td>
<td>Writing suffix separately</td>
<td>koṭiy-avān</td>
<td>vel-araī</td>
</tr>
<tr>
<td></td>
<td>(நீர்க்களை கூறுவது)</td>
<td>vel-araī</td>
<td>koṭi'-ōr</td>
</tr>
<tr>
<td></td>
<td>(பூச்சியற் வேளூர்)</td>
<td>cēvitaṅ</td>
<td>cēvitaṅ</td>
</tr>
<tr>
<td></td>
<td>(பேர்கள் போன்று)</td>
<td>titti'-il</td>
<td>titti'-il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>titti'-il</td>
<td>titti'-il</td>
</tr>
</tbody>
</table>
Thus various issues that need to be addressed while developing an OCR for Tamil-Brahmi script in order to recognize the Brahmi script meaningfully was discussed. These palaeographic and orthographic issues can be rectified by incorporating lexicon or Brahmi language database to OCR.

5. Conclusion

Many language recognition tools have been developed for modern and ancient languages. But, so far no standard tool was developed for Tamil-Brahmi script recognition due to the complexities of the script. In this paper, the various difficulties in developing an OCR for Tamil-Brahmi script was discussed. Further, the issues in terms of orthographical, palaeographical and morphophonemes were analysed in detail. In future, this analysis helps to develop linguistic rules that addresses the aforementioned issues thereby improves the accuracy of the Tamil-Brahmi OCR.

References

Vol I, Tamil Brahmi Inscriptions, Central Institute of Classical Tamil, Chennai.


புதியசாகவாக துறை திட்டப்பட் வருமான், 2020
சுவராத் 11, 12 & 13.

தரவத் துறையில் சிறந்தவுடன் பூர்வகம் அறிவு : கிளாப்புகள்
நிறைவு

துறையும், அறிவு
சுவராத்
சுவராத் தொடர்பான முறையில்

1. முன்வரை
தீர்த்ததுறையில் அதிக அளவில் இருந்த வருமானம் பலவகையான கருத்துறையின்
சுவரகம்பன் திரும்பத்தவாக தீர்த்ததுறையில் முன்வரை, பன்னாட்டு
பலவகையான கருத்துறையின் சுவரகம் கருத்தரம். இத்துறையில் குறான
இலைகளில் விளையாடி ஆட்சிகள் கருத்துறையில் சுவரகம் கற்றிக் கொட்டின, உதவி
பலவகையான கருத்துறையின் சுவரகம் கற்றிக் பெருந்து பதிவிட்டது.
அனைத்தும் கூறினாலான விளையாடி பலவகையான கருத்துறையின் பொழிப்பு, அறிய
தீர்த்ததுறையில் பூர்வகம் தொடர்பான முறையில் பொழிப்பு பொழிப்பு
பலவகையானது.

2. முறையும் தன்மை
முறையும் கருத்துறையில் பலவகையான கருத்துறையில் பலவகையானது. கட்டுரையும்
நேரிலும், சுவரையும், சுவரகம்பன், கருத்துறையில், பலவகையான
cொட்டிகளின் ஆராய்ச்சிகள் முழுநோக்கிட்டு கருத்துறையில் சுவரகம் கருத்தரம்.
தீர்த்ததுறையும் கருத்துறையில் கட்டுரையும் பலவகையானது. கட்டுரை
தீர்த்ததுறை 2017ஆம் ஆண்டில் உலகிலிருந்து ஒரு நாளுக்கு
தன்மையான நடப்பணிகள். அதிக அளவில் காணப்பட்டது. தீர்த்ததுறையில்
சுவரகம் பலவகைமுடிய அறிவு பலவகையான தீர்த்ததுறையில்
சுவரகம் முன்வரை உதவியுடன் அனைத்து காரணிகளாலும் தீர்த்ததுறை
சுவரகம் முன்வரை 2021ஆம் ஆண்டில் அதிக கருத்துறையில் சுவரகம்
சுவரகம் கருத்தரமின் மன்னரால் அவர்கள் அமைத்தர் அகத்துறை (பாற 1)
சுவரகம் முன்வரை.
1. குறிப்பிட்டுள்ள அறிக்கை

2. தரவக வழி

3. இரண்டாம் வளிமத

4. தமிழ் திட்டம் அறிமுகத்து
2. விகிதான் குற்றிகள் வழங்கத் தரும் அளி விதிகளின்றி குற்றிகள் வழங்கத்திறன் இணைக்காளாராலோலால், தரையும்புகழ் தன் குற்றிகள் வழங்கப்படுமிடத் தருக்கத்திறனை குற்றிகள் வழங்கத்திறன். விகிதான் விதிகளின் குற்றிகள் வழங்கத்திறன் பொழுதுபோக்கை பச்சைக்கிள்கில் கருதியுள்ள புதுக்குற்றிகளைக் கையாளாவிசித்து விளக்கிய விதிகள் விளக்கிய விதிகள் அக்கால புரோஸ்ஸிக்கால குற்றிகள் வழங்கத்திறன். விதிகள் விதிகள் வழங்கத்திறன் பச்சைக்குற்றிகளை கையாளாவிசித்து விளக்கும் நீர். விதிகள் விதிகள் வழங்கத்திறன் பச்சைக்குற்றிகளை கையாளாவிசித்து விளக்கும் நீர்.

புரோஸ்ஸிக்கால விதிகள் வழங்கத்திறன் பச்சைக்குற்றிகளைக் கருதியுள்ள புதுக்குற்றிகளை கையாளாவிசித்து விளக்கிய விதிகள் விளக்கிய விதிகள் அக்கால புரோஸ்ஸிக்கால குற்றிகள் வழங்கத்திறன்.

1. "வசாத்து அகராதி வசாத்து" { 'ஆம்' குறிப்பிட்டு அல்பான்பன்பிறைத்தை}:
2. {வசாத்து வசாத்து வசாத்து}
3. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
4. {வசாத்து வசாத்து வசாத்து}
5. 'ஆம்' குறிப்பிட்டு அல்பான்பன்பிறைத்தை: துறை
6. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
7. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
8. {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
9. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
10. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
11. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
12. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
13. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
14. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
15. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
16. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
17. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
18. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
19. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான์்}
20. மேல் 'காதல்' {வசாத்து வசாத்து வசாத்து குறிப்பிட்டு அல்பான்பன்பிறைத்தை}
21. வாழ்த்து
புதியசாரங்க இளை திசம்பரம் மார்ச் 2020
பகுதியான 11, 12 & 13.
உடல் உறுதி 1. கொண்டுவந்த வருவாயாக நிறுவாசிய
புத்தாண்டாம் பூத்தாண்டு தினசரி காலம், 2020

புத்தாண்டு பூத்தாண்டின் நிலைப்படி. தம்முள் சுருக்கமாக, தம்மும் முழு சோதனையில் ஆராய்ச்சி மாற்றமாக உள்ளது. இது குறுக்கு தரவு வாய்ந்து விளக்கமிட்டு ஸ்டாம்பத்தை வருபவற்கு பயன்படுத்துவது. இது தொடர்புள்ள வாய்ந்து விளக்க வாய்ந்து விளக்கத்திற்கு பதிவுபடுத்துவது தையும்.

6. முப்புறா

ஒவ்வொரு காலைக்குள்ளும் காலைக்குள்ளும் இருந்து பாதுகாப்புச் செய்யவும் காலைக்குள்ளும் பாதுகாப்புச் செய்யவும் ஆதாரங்களின் தொடர்பு என்று விளக்கியுள்ளன. காலை வெள்ளியை விளக்கும் விளக்கமாக விளக்கைக் காலை விளக்கியுள்ளன. காலை வெள்ளியை விளக்கும் விளக்கமாக விளக்கைக் காலை விளக்கியுள்ளன. காலை வெள்ளியை விளக்கும் விளக்கமாக விளக்கைக் காலை விளக்கியுள்ளன.
கோட்டா பல்கலைக்கழகத்தில். பல்கலைக்கழகத்தில் பயிரி பல்கலைக்கழகாட்கு பல்கலைக்கழகத் தமிழ் விளக்க விளக்கத்தில் விளக்கம் விளக்கத்தில் மூலவரின் கல்வி கல்வி பல்கலைக்கழகத் தமிழ் பல்கலைக்கழகாட்கு பல்கலைக்கழகத் தமிழ் விளக்க விளக்கத்தில் விளக்கம் விளக்கத்தில் மூலவரின் கல்வி கல்வி பல்கலைக்கழகாட்கு பல்கலைக்கழகத் தமிழ் விளக்க விளக்கத்தில் மூலவரின் கல்வி கல்வி பல்கலைக்கழகாட்கு பல்கலைக்கழகத் தமிழ் விளக்க விளக்கத்தில்.

முனைவர்: 
3. உ.ர. முதுவராஜ, ர. ராமசாமி, க. தெருசம், வ. பூணையா, க. தெருசம் “மூலவரின் கல்வி கல்வி அரங்கா” 2013 'சென்னையில் பல்கலைக்கழகத் தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக தமிழ் பல்கலைக்கழக 
Application of Letter Successor Varieties in Tamil Morphological Analysis

Dr.K.Rajan 1, Dr.V.Ramalingam 2, Dr.M.Ganesan 3
1 Dept. of Computer Engineering, Muthiah Polytechnic College
2 Dept. of Computer Science & Engineering, Annamalai University
3 Centre of Advanced Studies in Linguistics, Annamalai University
kaliyaperumalrajan@yahoo.co.in
aucsevr@yahoo.com
ganesan-au@yahoo.com
Annamalainagar. Tamil Nadu, India

Abstract

Morphological analysis of a word, in agglutinative and morphologically rich languages, is the prerequisite for processing text in higher levels. Models that are able to learn the morphology of natural languages are of great practical interest as tools for descriptive linguistic analysis and for minimizing the expert resources needed to develop morphological analyzers and stemmers. This paper is to evaluate the well-known letter successor varieties experimentally for Tamil morphology. The letter successor varieties (SV) and predecessor varieties (PV) are the statistical properties of a language corpus, which indicates the boundaries of morphemes. The segments identified in this process are used to construct stem and suffix dictionaries in addition to perform morphological analysis. These methods are trained on samples collected from the (Central Institute of Indian Languages) CIIL Tamil corpora. The performance of these approaches on morpheme segmentation are compared with the results of a rule based morphological analyser.

Keywords: Unsupervised Learning, Letter successor varieties, Tamil morphology, Morpheme Segmentation

1. Introduction

Morphological analysis of a word is the process of identifying the constituent morphemes of the word. This segmentation can be done at two levels in a written language text. One is word segmentation and the other is morpheme segmentation. Word Segmentation is an important problem in many natural language processing tasks such as speech recognition systems where there is no explicit word boundary information within a continuous speech utterance, or in interpreting written
languages such as Chinese, Japanese and Thai where words are not delimited by space. In other languages, words are combination of smaller meaning bearing units referred to as morphemes. The act of separating a word into its morphemes is called morphological analysis and/or morpheme segmentation. The morpheme segmentation algorithm attempts to find morpheme boundaries within word forms. These morphemes are classified into prefixes, stems and suffixes. Morphemes are used to identify words, which are semantically similar and improve the performance of the systems in document retrieval, document classification, machine translation and speech recognition.

Researchers in the area of data compression, dictionary construction, and information retrieval have all contributed to the automatic morphological analysis. One such approach proposes to identify morpheme boundaries first, and thus indirectly identify morphemes on the basis of the degree of predictability of the (n+1)th letter given the first n letters. This was first proposed by [1] and further developed by [2].

Commonly, algorithms designed for word segmentation utilize very little prior knowledge or assumptions about the syntax of the language. Instead prior knowledge about typical word length may be applied, and small seed lexicons are sometimes used for bootstrapping. The segmentation algorithms try to identify character sequences that are likely words, without the context of the words. Several approaches, based on machine learning, aiming at word segmentation and morphology have been published [3] [4].

2. Letter Successor Variety (LSV)

The idea of LSV is to count the number of different letters encountered after (or before) a part of a word and to compare it to the counts before and after that position. Morpheme boundaries are likely to occur at sudden peaks or increase of that value [38]. When the successor varieties for a given word have been derived, the information is used to segment the word. In [1] four different methods for performing segmentation are proposed as follows:
First method is the cutoff method in which a threshold is selected for boundary detection. The second one is peak and plateau method in which a segment break is made after a character whose successor variety exceeds that of its neighbors. The third method is complete word method in which a break is made if the segment is a complete word. The fourth method is based on the entropy value calculated for each letter in the word.

This work is based on the peak and plateau model. If \( S_n \) is the successor count of \( n^{th} \) character in a word, a word is segmented if \( S_n \) forms a local peak or a plateau of the count vector. It is also possible to segment the suffixes using the predecessor count \( P_n \), which is the count of number of different characters before the given suffix of length \( n \). This is estimated with the reverse of the test word and the reverse of the corpus. The words are represented in consonant-vowel form.

The motivation for using the successor varieties in the word segmentation task is the fact that, in a word, the \( I^{th} \) letter is dependent on the \( I-1 \) letters that precede it. Within a stem or suffix of a word, this dependence is quite strong and increases with increased \( I \). At the beginning of a new word, the dependence is considerably reduced. The successor variety is low within a word and tends to decrease from left to right. At the boundaries the successor variety increases. From the set of successor varieties and the peaks calculated for the test word, the boundaries of stems and suffixes can be detected.

Let \( W_n \) be a word of length \( n \); \( P_i \) is a prefix of length \( i \). Let \( C \) be a corpus of words. \( CP_i \) is defined as the subset of \( C \) containing those words whose first \( i \) letters match \( P_i \) exactly. The **successor variety (Succ_Variety)** of \( P_i \), denoted \( SV_i \), is the number of distinct letters that occupy the \( i+1^{st} \) position in words of \( CP_i \). The word to be tested of length \( n \) has \( n \) initial substrings and their corresponding successor varieties are \( SV_1, SV_2, \ldots, SV_n \). Near the beginning \( SV_i \) is high and near the end of long words, \( SV_i \) becomes very small, so that boundaries near the end of words go undetected. To avoid this situation the same process is repeated on reversed word and the reversed corpus, which returns the **predecessor variety (Pred_Variety)** counts [1]. This is also considered as successor variety in the reversed word list. For
each terminal substring of length \( n-i \), a predecessor variety count \( PV_{n-i} \) is determined.

Consider the following example that illustrates the LSV approach. Assume the word to be segmented is: பெத்தைணயமானை (’start’ - verb + Aux + present + 3\(^{rd}\)-person singular masculine - ‘he has started’)

Table 1 The successor and predecessor variety counts (From Tamil Corpus) with the segments made by peak values

<table>
<thead>
<tr>
<th>Starting Sequence</th>
<th>Ending Sequence</th>
<th>Succ_Variety</th>
<th>Pred_Variety</th>
<th>Segment SV</th>
<th>Segment_PV</th>
<th>Stem</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
<tr>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>பெத்தைணயமானை</td>
<td>பெத்தைணயமானை</td>
</tr>
</tbody>
</table>

The Table 1 shows the successor and predecessor variety counts with the segments made by the peak values. In the above example, the successor variety of first character is always higher because of the more the number of words start with that single character. It is ignored as the stem has only single character. The starting sequence “பெத்தைணய” has a peak successor varieties count, and marks the segment boundary. The starting sequences பெத்தைணயமானை, பெத்தைணய, பெத்தைணயமானை, பெத்தைணயமானை and பெத்தைணயமானை have peak values when compared to their neighbours. So, they mark 5 different stem boundaries. When combining these boundaries, the result produces the
morphological analysis from left to right. The ending character sequences அ, உந்தம், தமிழ்நாடு and பரப்பல்கள் have peak predecessor variety counts. So, they form the suffix boundaries.

The successor varieties (SV) of the following substrings form peaks which produce stem boundaries.

உந்தம் followed by {தமிழ்நாடு, தமிழ்நாடு, பரப்பல்கள், பரப்பல்கள், Space} with count 5

உந்தம் தமிழ்நாடு followed by {அ, உ, எ, ச, ப, எ} with count 7

உந்தம் தமிழ்நாடு தமிழ்நாடு followed by {அ, உ, எ, ச, ப, எ, உந்தம்} with count 4.

உந்தம் தமிழ்நாடு தமிழ்நாடு தமிழ்நாடு followed by {அ, உ, எ} with count 3

உந்தம் தமிழ்நாடு தமிழ்நாடு தமிழ்நாடு தமிழ்நாடு followed by {அ, உ, எ} with count 3 in the selected corpus

The Highest successor count occurs at the beginning of a word. The letter ‘அ’ has 10 different successor characters. This single character stem is not considered as the real segment. The boundaries mark the stem and suffix segments as listed below.

All the segments can be combined and represented as morphological segmentation.

From the above segments, it is observed that the first segment produces stem of a given word form.
The **Letter Predecessor Variety (LPV)** counts are estimated as the number of different characters preceding the *n-i* terminal characters (specified as suffix). The predecessor variety counts are usually more because there is a greater number of words which have the given terminal strings as suffixes. Successor and predecessor counts are stabilized when the corpus has examples of different inflections and conjugations. Predecessor counts for suffixes are usually higher than successor counts for stems. The Figure 1, and Figure 2 show the peaks of successor variety counts and predecessor variety counts for letter sequences respectively.
3. Data Set and Segmentation

3.1 Data set

The letter successor varieties method is performed on the following data set. The CIIL (Central Institute of Indian Languages) corpus of 3 million words is used for the analysis. It consists of 750 files belonging aesthetics, fine arts, natural science, social science and official and media language. The text files are tokenized and the resulting words are preprocessed by removing punctuations, special character sequences and ending hard consonants. The words of length less than 3 characters and greater than 30 characters, and function words which have very high frequency are removed. The shorter words and function words do not have inflections and segmentation is not required. The unique word list is prepared. There are more than 4,000,000 unique words. From these words a subset of 16,148 words are used for successor variety analysis. The words are represented as a sequence of consonants and vowels. So, each character will take only one byte. The words are segmented into two parts as shown in Table 1.
3.2 Segmentation

The segmentation of stem and suffixes is performed from the measured successor counts. The segmentation is made in the test word, where one or both of the successor and predecessor varieties forms peak. The Table 1 shows the boundaries marked in the segment SV (made from SV) and segment PV (made from PV) columns. The boundary at position i is marked as 1, if the corresponding variety count is greater than their neighbours.

\[
\text{Segment}_{SV}i = 1, \text{if } (SV_i > SV_{i-1} \text{ and } SV_i > SV_{i+1})
\]

\[
\text{Segment}_{PV}i = 1, \text{if } (PV_i > PV_{i-1} \text{ and } PV_i > PV_{i+1})
\]

The segment with more than two characters in prefixes marked as 1 in Segment SV is the actual stem. It is also observed that, the stem alternant has also been marked. This segmentation produces list of stems and suffixes. The logical AND and OR operation of segment SV and segment PV are also considered for performance analysis. The successive boundaries mark the morpheme boundaries and results in morpheme segmentation. The list of stems and suffixes can be collected and stored in the lexicon. The following criteria are used for determining the boundaries from the estimated count values.

**Cutoff for successor count and predecessor count:** One way to segment the word is to set some cutoff K and then break the word where the variety counts reaches or exceeds K. The results from experiments show that the cut off for successor and predecessor counts cannot be fixed at the optimum level, as they vary widely. This cut off measure is not used for segmentation in this work.

**Successor count at peak:** This segmentation breaks the words at the peak of the successor variety counts. This technique produced high accuracy for stem identification.

**Predecessor count at peak:** The segmentation based on the predecessor counts produce more morphemes at near the suffixes.
The logical AND of successor and predecessor peaks: The segmentation (break) is made only when both the quantities are at peak. The successor and predecessor counts individually mark many false cuts. It is unlikely that the same false cuts will be found by both counts. So, the precision of this method is higher than that of the individual peaks and the recall is low.

The logical OR of successor and predecessor peaks: In this experiment both the peaks of successor and predecessor counts are used. A break is made either at successor peak or predecessor peak. This method increases the number of cuts made and the recall is also increased.

4. Evaluation of Segmentation

Word and morpheme segmentations are typically evaluated in terms of the accuracy and coverage of the proposed word or morpheme boundaries. In order for a word or morpheme to be correct, its two delimiting boundaries must naturally be correct. The evaluation method consists in comparing the proposed morpheme boundaries to linguistic gold standard segmentation. As mentioned above, such an evaluation is straightforward and intuitive, provided that an adequate gold standard exists. Segmentation gold standards are valuable resources, which require large amounts of work by linguistic experts. For our research we prepared the gold standard segmentation for the entire corpus. Once the gold standard exists it can be used for the evaluation of unsupervised learning algorithms. Total Number of Segments in GOLD standard segmentation: 63108 obtained from 16148-word forms.

Precision and recall

Precision is the proportion of correct boundaries among all morpheme boundaries suggested by the algorithm. Recall is the proportion of correct boundaries discovered by the algorithm in relation to all morpheme boundaries in the gold standard. Given a reference segmentation, precision (how many of the predicted segmentation points were correct) and recall (how many of the correct segmentation points were found) can be calculated.
True Positive (TP) refers to the correct morpheme boundaries.

False Positive (FP) refers to the wrong boundaries in the predicted segmentation that is morpheme boundary being predicted at location where no morpheme boundary exists.

False Negative (FN) refers to the missing correct boundaries in the predicted segmentation.

\[
Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (1)
\]

\[
Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (2)
\]

\[
F \text{ Measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)
\]

Table 2 Sample morpheme segmentations

<table>
<thead>
<tr>
<th>No.</th>
<th>Over all Word Segmentation</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>சுருக்குமரப் தமிழ் மற்றும் இணைணயம் (Reference or Gold Segmentation)</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>சுருக்குமரப் + தமிழ் + இணைணயம் + இடைய (SV)</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>சுருக்குமரப் + தமிழ் + இணைணயம் + இடைய (PV)</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>சுருக்குமரப் + தமிழ் + இணைணயம் + இடைய (SV AND PV)</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>சுருக்குமரப் + தமிழ் + இணைணயம் + இடைய (Rule based)</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The Table 2 shows how the performance of the segmentation is estimated using TP, FP and FN with the equations 1 and 2. The accuracy is the percentage of words whose proposed segmentation is identical to the correct segmentation. The F-measure is the harmonic mean of precision and recall. It is estimated using equation 3. To evaluate the performance of LSV segmentation algorithm precision, recall and F-measure are used.

LSV algorithm performs well on stemming. Without any rule base, the stemming can be performed from the sufficiently large corpus. LSV and LPV are
more suitable for handling concatenative morphology (Eg. Tamil) than other types of morphology. Corpus containing contextually similar words which share grammatical information with the input word is required for better performance. The words having many forms in the corpus have higher accuracies. The performances of LSV and LPV with different segmentation strategies are tabulated in Table 3. They are compared with results of Rule based morphological analyser.

5. Performance Comparison of Various Methods on Morpheme Segmentation

The performances of various unsupervised segmentation algorithms are compared. The number of segments made by these methods are compared with the actual segmentation points given in the Gold Standard segments. The precision, recall and F-measure are estimated and tabulated in Table 3 and graphically shown in Figure 3.

Table 3. Performance of different methods on morpheme segmentation

<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision %</th>
<th>Recall %</th>
<th>F-Score %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Succ_Variety_Peak (SV)</td>
<td>80.74</td>
<td>68.34</td>
<td>74.00</td>
</tr>
<tr>
<td>Pred_Variety_Peak (PV)</td>
<td>72.23</td>
<td>74.18</td>
<td>73.19</td>
</tr>
<tr>
<td>Peak of Both SV AND PV</td>
<td><strong>86.99</strong></td>
<td>65.78</td>
<td>74.92</td>
</tr>
<tr>
<td>Peak of SV OR PV</td>
<td>66.02</td>
<td><strong>76.57</strong></td>
<td>70.91</td>
</tr>
<tr>
<td>RULE BASED</td>
<td><strong>95.63</strong></td>
<td>48.94</td>
<td>64.75</td>
</tr>
</tbody>
</table>

The precision of the logical AND of SV and PV has comparatively higher precision than other SV and PV models, but the rule-based segmentation performs with 95.63% of precision. The logical OR of SV and PV return more segments, which results in increased recall of 76.57%. The low recall of rule-based system is due to the absence of results for some words. The words with auxiliary morphemes are not analysed by the system. The F-measure of the methods indicate that the intersection of SV and PV peaks produce higher value which is 74.92% among the LSV approaches.

The performance reduction in the LSV methods is characterized by the following:
The computed stem is a substring of the true stem. The long true stems are divided at the boundaries of short stem boundaries.

Short words cause incorrect segmentation. In the literature only the words of 5 or more letters are considered for segmentation.

The stem alternant forms and allomorphs of various morphemes results in low precision. [In Tamil, there are more allomorphs for some categories]

Rare word forms and words with small number of inflections are not segmented properly.

![Performance on Morpheme Segmentation](image)

**Figure 3. Performance Comparison of unsupervised methods (SV-PV) and rule-based method on morpheme segmentation**

**Conclusion**

This empirical study performed various combination of letter successor and predecessor varieties on Tamil corpus and evaluated the performance of those methods on morpheme segmentation. The results are compared with the outputs of rule based morphological analyser. The results show that accurate morpheme segmentation (analysis) is achieved by this process without using any dictionaries or rules. From the segmentation output, the characteristics of morphemes, allomorphs and suffix patterns can be studied.

**References**

[5]. M. Ganesan, “Morph and POS Tagger for Tamil” (Software), Annamalai University, Annamalai Nagar, 2009.
Algorithm to Correct Missing *Pulli*-Signs in Printed Tamil Text

Author: Muthiah Annamalai*

**Abstract:**
A common problem in digitizing Tamil texts is the missing consonant sign *pulli* (unicode u0BCD) which may be lost in the process of OCR, or absent by convention during printing of older texts. We propose a bigram statistics based combinatorial algorithm to correct such errors in text.

1. **Introduction**
A common problem in digitizing Tamil texts [1] is the missing consonant sign *pulli* (unicode u0BCD [5]) which may be lost in the process of OCR, or absent by convention during re-printing of older texts; in one case the reprints contain newer errors than in original library copies. We propose a bigram statistics based combinatorial algorithm to correct such errors in text, adding to the classes of algorithms for Tamil text correction presented previously [4].

2. **Methodology**
A given Tamil word is split into Tamil letters and we consider to correct only the missing *pulli* or consonant sign. Since *pulli* can go with only the consonants, we find each consonant occurring in the word can be true consonant or a false consonant yielding at least one correct alternate in the 2 raised to power of number of consonants in the word.

For example, a misprinted word கணணன் (correct form is only 'கணன்'), has 4 consonants [க, ண,ண, ன]. Therefore we may expect upto 16 variants of the word with and without the pulli, at least one of which will be the correct answer.

```plaintext
['கணணன்', 'கணணன்', 'கணணன்', 'கணணன்',
'கணணன்', 'கணணன்', 'கணணன்', 'கணணன்',
'கணணன்', 'கணணன்', 'கணணன்', 'கணணன்',
'கணணನ்', 'கணணன்', 'கணணன்', 'கணணன்']
```

The algorithm for generating the combinatorial alternates is shown in code `pulligal_helper` in Appendix A. The complexity of this algorithm is $O(2^{\text{consonants}})$.

**Algorithm:**

**Input:** Tamil word $W_r$ with potentially one or more missing pulli(s)
Output: Best alternative word \( W_c \)

1. Let \( C \) be list of all the consonants in order of occurrence in \( W_r \)
2. Generate set \( S \) of all words with consonants \( C \) occurring with and without pulli
3. Since there is only the binary choice - i.e. each consonant can occur with or without pulli we have \( 2^{|C|} \) alternatives
   a. These alternatives can be simply generated by looking at bit pattern of enumeration of numbers from \((0, \ldots, 2^{|C|}-1)\).
   b. For each of the bit pattern we can associate 1 with presence and 0 with absence of pulli sign.
4. Sort the alternatives by their bigram probabilities; e.g. bigram probability of occurrence of a \( n \)-letter word in a language can be written as, with base bigram probabilities, \( P(w_i|w_{i-1}) \), picked off a standard reference - e.g. Tamil VU Dictionary or Project Madurai [2a,2b,4].
   a. \( P(W_n) = \prod_{i=1}^{n} P(w_i|w_{i-1}) \)
5. A reasonable alternative has highest probability of occurrence and can be marked as output word.
6. Optionally we can combine step 5 by filter all words \( S \) for a prefix occurring in a dictionary \( D \).

Once we generate this list of alternates we can filter them with a dictionary or a unigram or bigram scores. The bigram score is computed using the Tamil Virtual University dictionary based corpus statistics in Open-Tamil library[2a,2b]. Bigram or unigram probabilities for the word are computed in standard manner using the sum of logarithm of individual bigram probabilities. The computational complexity of the algorithm is exponential in number of consonants occurring in the word but usually this is of order of 1-10 consonants even in a large word.

3. Results
The Sangam poetry lines by classical Tamil poet Chembulapeyalneerar [3] when misprinted (without the pulli) appears as follows:

<p>| Incorrect | பாரமுத் தேமும் பாரகம்பாக இராச்சும் துறலையும் கைற்று இருந்தது |
| Correct   | பாரமுத் தேமும் பாரகம்பாக இராச்சும் துறலையும் கைற்று இருந்தது |</p>
<table>
<thead>
<tr>
<th>Misprinted Word</th>
<th>Alternates</th>
<th>Bigram Scores</th>
<th>Correct Word</th>
</tr>
</thead>
</table>
| பாமும் | 'பாமும்'  
           'பாமும்' | 20            | 16  | பாமும் |
| மதாவுரு | 'மதாவுரு'  
           'மதாவுரு' | 20            | 16  | மதாவுரு |
| மாநாயிட்டுள்ளாகியேரா | 'மாநாயிட்டுள்ளாகியேரா'  
                     'மாநாயிட்டுள்ளாகியேரா' | 36            | 31.94  | மாநாயிட்டுள்ளாகியேரா |
| மாமறுமையும் | 'மாமறுமையும்'  
           'மாமறுமையும்' | 32            | 28.83  | 28  | 24.83  | மாமறுமையும் |
| தமிழ் | 'தமிழ்'  
           'தமிழ்' | 36            | 32.839  | 23.0  | 28.839  | தமிழ் |
| சம்பாகத்து | 'சம்பாகத்து'  
           'சம்பாகத்து' | 32.18 | 28.54  | 28.18  | 24.54  | சம்பாகத்து |
| சோழநிதி | 'சோழநிதி'  
           'சோழநிதி' | 20            | 16  | சோழநிதி |

In total we see just a bigram based scoring of the 7 misprinted words give correct choices for 6 words of 7 in error, for an 85.7% accuracy score, showing the promise of our approach. Further improvements in the accuracy can be made by using a dictionary based approaches - for example in sub-routine `pulligal_branch_bound` in the Appendix A.

4. Conclusion
We present a simple algorithm for correcting OCR induced missing consonant sign by combinatoric generation and filtering by bigram scoring. This approach is useful for advanced spelling correction on lines of our work in [4].
5. References

   (b) Soltiruthi module of Open-Tamil v0.96 https://pypi.org/project/Open-Tamil/ (accessed Jun, 2020).
Appendix - A

import tamil
import operator
from solthiruthi.scoring import bigram_scores, unigram_score

def mean(x):
    return sum(x)/float(len(x))

def pulligal_helper(prefix,letters):
    if len(letters) == 0: return [prefix]
    letter = letters[0]
    result = []
    if letter in tamil.utf8.agaram_letters:
        result1 = pulligal_helper(prefix + letter, letters[1:])
        mei_letter = letter + tamil.utf8.pulli_symbols[0]
        result2 = pulligal_helper(prefix + mei_letter, letters[1:])
        result.extend(result1)
        result.extend(result2)
    else:
        result1 = pulligal_helper(prefix + letter, letters[1:])
        result.extend(result1)
    return result

def pulligal_branch_bound(prefix,letters,அகராதி):
    """ we restrict options if its not a prefix in dictionary """
    if len(letters) == 0: return [prefix]
    letter = letters[0]
    result = []
    prefer = அகராதி.starts_with(prefix)
    if letter in tamil.utf8.agaram_letters:
        alternate2 = prefix + mei_letter
        if அகராதி.starts_with(alternate2) or prefer:
            mei_letter = letter + tamil.utf8.pulli_symbols[0]
            result2 = pulligal_branch_bound(alternate2, letters[1:])
            result.extend(result2)
        alternate1 = prefix + letter
        if அகராதி.starts_with(alternate1) or prefer:
            result1 = pulligal_branch_bound(alternate1, letters[1:])
result.extend(result1)
return result

#sort in descending order
chol = tamil.utf8.get_letters(input("Enter Word>>>"))
result_tpl = [("".join(sol),-1.0*bigram_score(sol)) for sol in pulligal_helper("",chol)]
result_tpl = sorted(result_tpl,key=operator.itemgetter(1),reverse=True)
print(result_tpl)
1. Introduction

A morphologically complex word containing at least two elements which can otherwise occur as free forms (i.e. as independent words) can be considered as a prototypical compound. e.g.

1. talaiyaNai (< talai ‘head’ + aNai ‘support’) ‘pillow’
2. marappeTTi (< maram ‘wood’ + peTTi ‘box’) ‘wooden box’
3. kiLippaccai (< kiLi ‘parrot’ + paccai ‘green’) ‘parrot green’

Compounding is a grammatical process by which complex words are formed from smaller elements which have word status under normal circumstances. Affixation is different from compounding as it involves morphemes which do not have word status. It is the word-like behaviour of a string of elements which indicates that it is a compound. Though rearrangement of constituents in a construction is possible in a language, the constituent parts of compounds cannot be rearranged. e.g.

1. talaiyaNai ‘pillow’ vs. ?aNaitalai
2. marappeTTi ‘wooden box’ vs. ?peTTimaram
3. kiLippaccai ‘parrot-green’ vs. paccaikkiLi ‘parrot’

(“?” mark indicates that the expression does not mean or refer anything.)

In addition, the constituents of a compound do not allow themselves to be separated by intervening material showing word like quality. In languages like English the compounds are distinguished from phrases by their typical stress pattern. In some languages there is linking morpheme compounding the constituents (as found in the compound morph-o-syntax).

The compounds can be studied at least from five points of view:

1. Based on the grammatical categories of words which constitute compounds
2. Based on the semantic classes
3. Based on the possible linking elements
4. Based on the deep structure
5. Based on the morphophonology.

Compounds which are nouns are called nominal compounds. Traditional grammarians of Tamil look at nominal compound formation in Tamil from two points of view:

1. Compound nouns are derived from phrases by the deletion of elements like case suffixes, comparative particles, tense suffixes, co-ordinate particles, and predicative elements.
2. Compound nouns are derived only by the juxtaposition of words and only for the interpretation of meaning they have to be expanded with the help of elements mentioned above.
The modern approaches, whether they represent lexicalist approach or generative approach, are not totally different from the traditional view points on nominal compounds.

2. Distinction between the root compounds and synthetic compounds

Distinction has been made in the literature between primary (root) compounds and synthetic (verbal) compounds. Primary compounds are simply concatenated words. e.g.

kaaTu 'forest' + vilangku 'animal' = kaaTTuvilangku 'wild-animal'

Synthetic compounds are formed from deverbal heads and non-heads which fulfil the function of the argument of the verb from which the head is derived. e.g.

vaNTi 'cart' + ooTTi 'driver' = vaNTiyooTTi 'cart driver'

(ooTTu 'drive' + i = ooTTi `driver'; -i is an agentive suffix.)

formation of root compounds and synthetic compounds and both are formed by the rule of the following:

N + N → N

The fact that the second head noun is derived from a verb will be taken care of by the word-formation of rule of the following type:

V + Suffix → N

There are a number of approaches: Traditional approaches, Levi’s approach, Vijayavenugopal’s approach, etc. They have been discussed elaborately in Rajendran (1997, 2004, 2005) Due to want of space they are discussed here. We have taken up Knowledge Based Representation approach which is the core of our paper.

3. Knowledge Based Representation approach

As we have mentioned already that the position taken up by us is one of interpretation rather than generation from deep structures and that the process of nominal compound formation can be simply denoted by the rule, N + N → N, the computer analysis we propose here aims to interpret the meaning of the compounds from the information available in the nouns participating in the compound formation. We can take relevant clues from traditional grammars, Levi (1978) and Venugopal (1979). The representations of nominal compounds to interpret their meanings by the above mentioned three approaches show us that there are relations existing between the concatenated nouns which are to be established through certain verbs or predicates. That means we have to look forward for a grammatical formalism which can help us to interpret or retrieve the unexpressed part linking the constituents of a compound from the information available in the constituents themselves.

A knowledge-based system will serve our purpose. “Knowledge-based system emphasise meaning. Instead of processing data as a string of bits, they represent the meaning of data in terms of the real world. They carry on conversations with people in ordinary language; they find important facts before
they are requested, and they solve complex problems at expert level of performance.” (Sowa, 1983: Preface). Artificial intelligence (AI) and cognitive science are the two fields which are devoted to knowledge-based systems. Cognitive science takes into its fold philosophy, linguistics and psychology allowing a strong influence from computer science. Artificial intelligence is the engineering part and it focuses on programming tools and techniques than to philosophical issues. The present analysis makes use of both cognitive science and artificial intelligence. Conceptual graphs which emphasise meaning are widely used in AI systems.

Broadly speaking conceptual graphs are logical forms that state relationships between entities, attributes, and events. A detailed study of the notion of conceptual structure adopted here is available in Sowa (1984). The conceptual relations are similar to case relations established by case oriented grammars. The sentences *A man bites a dog* and *A cat is sitting on a mat* can be represented by conceptual graphs as follows:

![Conceptual Graphs](image)

The information regarding tense and modalities are omitted in the above graphs. The square brackets enclose concepts and the braces enclose conceptual relations. “(AGNT)”, “(OBJ)” and “(LOC)” denote respectively the conceptual relations (similar to case relations) AGENT, OBJECT and LOCATION. Though the conceptual relations are finite in number, it is not attempted here to establish the whole set of conceptual relations for Tamil as it will take us to a different direction. The conceptual relations which are needed for the interpretation of compound nouns are already in vogue in case oriented grammars and are self-explanatory.

There are at least three levels of complexity of conceptual graphs which we have to account here:

1. Arbitrary conceptual graphs which impose no constraints on permissible combinations.
2. Canonical graphs which enforce selectional constraints. They correspond to the case frames in linguistics and the category restrictions in philosophy.
3. Schemata which incorporate domain-specific knowledge about the typical constellations of entities, attributes, and events in the real world.

“A canonical graph is a combination of concept nodes and relation nodes where every arc of every conceptual relation is linked to concept.” (Sowa, 1984:90). As not all such combinations make sense and some of them include absurd combinations
like the following (which is an outcome of Chomsky’s famous example *Colourless green ideas sleep furiously.*) certain graphs are declared canonical:

$$\text{[IDEA]} \leftarrow \text{(AGNT)} \leftarrow \text{[SLEEP]} \rightarrow \text{(MANN)} \rightarrow \text{[FURIOUSLY]}$$

$$\downarrow$$

$$\text{(ATTR)}$$

$$\downarrow$$

$$\text{[COLOURLESS GREEN]}$$

Canonical graphs are meaningful graphs which represent real or possible situations in the external world. Schema is the basic structure for representing background knowledge for human like inference. Schemata favour plausible combinations by incorporating more knowledge about the world and avoiding less likely possibilities. While canonical graph represent everything that is conceivable, schemata represent everything that is possible. The following is the schema for BUS (Sowa, 1984:129):

$$\text{[BUS:}*x\text{]} \leftarrow \text{(INST)} \leftarrow \text{[TRAVEL]} \rightarrow \text{(RATE)} \rightarrow \text{[SPEED} \leq 55\text{mph]}$$

$$\uparrow$$

$$\downarrow$$

$$\text{(CONT)}$$

$$\downarrow$$

$$\text{[OBJ]}$$

$$\uparrow$$

$$\downarrow$$

$$\text{[DRIVE]}$$

$$\downarrow$$

$$\text{[PASSENGER:}{*}\text{]}$$

$$\downarrow$$

$$\text{(AGNT)}$$

$$\downarrow$$

$$\text{(QTY)}$$

$$\downarrow$$

$$\text{[DRIVER]}$$

$$\text{[NUMBER} \equiv 50\text{]}$$

(In the above diagrammatic representation of schema for BUS, “*x” is a variable in which “*” is a generic marker and “x” is an indicator for cross reference, “(INST)” denotes the relation INSTRUMENT, “(CONT)” denotes the relation CONTAIN, “(QTY)” denotes QUANTITY, “≤” denotes ‘lesser than or equal to’ and “≡” denotes range.)

4. **Computational analysis of Nominal Compounds**

Our proposal for automatic interpretation of meaning of the compounds from the meanings of concatenated nouns leads us to turn our eyes to an approach founded on cognitive science and artificial intelligence of knowledge representation. Conceptual graphs will be used to serve our purpose. From the meanings of constituent nouns (which
constitute a compound) which are expressed in terms of conceptual graphs, it is possible for us to establish the meaning of the compound.

Take for example the compounds *ndooy kirumi* ‘germs which causes disease’. A schema of *kirumi* = GERMS should contain the information that ‘germs can cause disease’ and similarly a schema of *nooy* = DISEASE should contain the information that ‘disease can be caused by germs’. That means the schemata for both the constituent nouns contain the following common graphical portion which might have been expressed differently:

\[
\text{[GERMS]} \leftarrow \text{(AGNT)} \leftarrow \text{[CAUSE]} \rightarrow \text{(OBJ)} \rightarrow \text{[DISEASE]}
\]

The unification of these two graphical portions can give us the following conceptual graph for the compound *ndooy kirumi*:

\[
\text{[DISEASE CAUSING GERMS]} \leftarrow \text{(AGNT)} \leftarrow \text{[CAUSE]} \rightarrow \text{(OBJ)} \rightarrow \text{[DISEASE]}
\]

The schema for the *kaakkaay* = CROW and *kuuTu* = NEST should contain the information that ‘crow can build nest’ and ‘nest can be built by crow’ respectively. The following graphical portion will be shared by both the constituent nouns:

\[
\text{[NEST]} \leftarrow \text{(OBJ)} \leftarrow \text{[MAKE]} \rightarrow \text{(AGNT)} \rightarrow \text{[CROW} \subset \text{BIRD]}
\]

“ (⊂) “ should be read here as ‘sub-set of’)

The meaning of the compound *kaakkaaykkuuTu* can be interpreted by the unification of the relevant portions of the schemata. The schema for *maN* = CLAY as well as *pommai* = TOY should contain the information that ‘clay can be used to make toy’ and ‘toy can be made up of clay’ respectively. The following graphical portion will be shared by both the constituent nouns:

\[
\text{[TOY]} \leftarrow \text{(OBJ)} \leftarrow \text{[MAKE]} \rightarrow \text{(AGNT)} \rightarrow \text{[MAKER]}
\]

\[
\downarrow \quad \text{(MADE UP OF)}
\]

\[
\text{[CLAY} \subset \text{MATERIAL]}
\]

The meaning of the compound *maN pommai* can be interpreted by the unification of the relevant graphical portions of the schemata of the constituent nouns.

There is no need of cataloguing of schema for each concept as the schema of a concept at a higher level can be used to build the schema for a concept at lower level. For example the schema for BIRD can be used to build schema for CROW and *kuruki* = SPARROW and also the interpretation of *paRavaikkkuuTu* ‘bird’s nest’ can take care of
the interpretation of *kaakkaik kuuTu* ‘crow’s nest’ and *kuruvikkuuTu* ‘sparrow’s nest’.
The hierarchical representations of canonical graphs can be captured by having a
thesaurus in which the concepts are arranged in the hierarchical fashion. The following is
a tesauric model for PHYSICAL OBJECT:

The computational process of interpreting the meaning of nominal
compounds from the meanings of the nouns which constitute them can be depicted
by the flow chart given below:

```
<table>
<thead>
<tr>
<th>RULES OF PARSING</th>
<th>PARSING OF NOMINAL COMPOUND INTO BIPARTITE UNITS OF NOUNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>CATALOGUE OF SHEMATA OF NOUNS</td>
<td>SELECTION OF RELEVANT SCHEMATA</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RULES OF PATTERN MATCHING</td>
<td>FINDING OF IDENTICAL OR SIMILAR GRAPHICAL PORTIONS</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RULES OF EXTRACTION OF IDENTICAL OR SIMILAR GRAPHS FROM SCHEMA</td>
<td>EXTRACTION OF IDENTICAL OR SIMILAR GRAPHICAL PORTIONS FROM THE SCHEMATA</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RULES OF UNIFICATION OF GRAPHS</td>
<td>UNIFICATION OF THE GRAPHS INTO A NEW GRAPH</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RULES OF CONVERTING GRAPHS INTO LOGICAL FORM</td>
<td>CONVERTING OF THE NEW GRAPH INTO LOGICAL FORM OF SEMANTIC REPRESENTATION</td>
</tr>
</tbody>
</table>
```

Algorithm:

1. Parse the nominal compounds into bipartite units of constituent nouns.
2. Select the schema for each noun which constitute the compound from the conceptual catalogue of schemata.

3. Identify the similar or identical graphical portions of schemata.

4. Extract the identical portions.

5. Unify them in such a way that the head-nounship is not altered.

6. Covert the conceptual graph into logical proposition which represent the meaning of the compound.

5. Conclusion

The proposed alternative of automatic interpretation of the meaning of nominal compounds from the meanings of nouns which constitute them is aimed to recapitulate the psychological reality of understanding by human brain in which the percepts are stored as conceptual graphs and the building up of and understanding of the new complex graphs is based on the already existing or stored conceptual graphs. Though knowledge representation by conceptual graphs is a tedious process and basing the semantic interpretation of compounded lexical units on the graphical representations of concepts is a challenge and sometimes may appear as a futile task, such attempt will depict the psychological reality of understanding by means of the information stored in the brain in terms of perceptual models called conceptual graphs. The deep structure representations from which the compounds are derived by transformational rules or formative rules is artificial and analysing and understanding a derived language unit by making use of a formalism whose foundation is deep structure representation does not represent a model which tries to capture the cognitive process of understanding and generation of language units. As conceptual graphs are finite they can be easily stored and manipulated by the computer.

The conceptual semantic interpretation of nominal compounds helps in the translation of the compounds into other language. This type of conceptual graphic representation lends a helping hand in the machine translation in which the interlingual representation of compounds is a challenging job.

REFERENCE


TamilInayaVaani - Integrating TVA Open-Source spellchecker with Python

Authors: T. Shrinivasan, Nithya Duraisamy, Ashok Ramachandran, Manickkavasakam, Arunmozhi, and A. Muthiah
Email: tshrinivasan@gmail.com, nithyadurai87@gmail.com, ashokramach@gmail.com, manikk.h@gmail.com, aruntheguy@gmail.com, and ezhillang@gmail.com

Abstract
We report the integration of Tamil Virtual Academy (TVA) Open-Source publication of TamilInaya Vaani spell checker into the existing Python environment for Tamil NLP applications, uses. Significant modification from the baseline spell checker was carried out including porting to Python, integration with Tamil sandhi checker, python packaging and addition of web-interface. We report limitations of this spell checker and future directions for this project.

1. Introduction

There are several open-source spelling checkers for Tamil with varying levels of completeness and in order of wider use we list them as, Hunspell, GNU ASpell, Langtool, Open-Tamil Solthiruthi, TamilSpell Checker with BloomFilter [6-10], and now we introduce TamilinayaVaani by this work.

Part of technical problem of correcting Tamil spelling errors in text is due to discrete infinity of vocabulary generation mechanisms in the morphologically rich agglutinative language, conjugation “sandhi” and free-word order properties of Tamil language. This prevents us from seeing much success while applying cookie-cutter techniques like Levenshtein-distance search on a fixed wordlist either exhaustively, a poor choice, or via selective search of edit distance by the Norvig algorithm.

Applying case rules, and stemming the words and using list of commonly occurring errors have shown much success in fairly proprietary spelling checkers like Vaani, MenThamizh (Amma Tamil), etc. which are current state of the art. However, these spelling checkers still have their own limitations including searching for 1-edit distance errors and limited in application by lack of concordance and word-sense disambiguation features to have robust suggestion algorithm.
Recently, after sustained efforts by our collaborators, the Tamil Virtual Academy had released source code of its commissioned project for Vaani Desktop Spell Checker under terms of GPL (v2) license. This allows open-source work on the code despite the limitations of obfuscated nature of the source, and undocumented nature of the spelling checker at time of release.

Our contributors ported the C# version of TamilnaiyaVanni spellchecker to Python language. This provides tons of new opportunities to mix with their own applications and make new desktop applications, plugins to other systems, web and mobile applications.

We converted the python code to a python module. Added a sandhi checker available in Open-Tamil python library and released as a new python module - `tamilinayavaani`. Apart from the python version, a javascript port of the same spellchecker is also in progress.

2. Organization of Python Spell Checker
Fig. 1. a Organization of Python module tamilinayavaani; the REST interface portion is outside the module but a static method serves as callback handler for the web server.

3. Integration with Sandhi-Checker

Tamil, along with Sanskrit, have concept of sandhi and specific rules on repetition, fusion and exclusion of letters (from a normative form of the word) when two words appear together in context their spelling is altered.

Tamil Sandhi Checker [3] published in 2018, is available as python module “tamilsandhi” distributed along with Open-Tamil package. This contains over 40 rules which enable improved quality of results.

4. Python Packaging

This project library can be installed as a Python package tamilinayavaani [5] using the commands:

```bash
$ python3 -m pip install --upgrade tamilinayavaani>=0.13
```

This provides two interfaces, namely for use in-memory and for use on a disk-file as follows,

<table>
<thead>
<tr>
<th>Usage Type</th>
<th>Code Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Memory Usage</td>
<td><code>from tamilinaiyavaani import SpellChecker</code></td>
</tr>
<tr>
<td></td>
<td><code>flag,suggs=SpellChecker.REST_interface('வாைழ', 'பழ#')</code></td>
</tr>
<tr>
<td></td>
<td><code>expected=['வாைழ%', 'பழ#']</code></td>
</tr>
<tr>
<td></td>
<td><code>assert not flag</code></td>
</tr>
<tr>
<td></td>
<td><code>assert expected[0]==suggs[0]</code></td>
</tr>
<tr>
<td>File based Usage</td>
<td><code>from tamilinaiyavaani import SpellChecker, SpellCheckerResult</code></td>
</tr>
<tr>
<td></td>
<td><code>result = SpellChecker(fname).run()</code> #fname is a full filename</td>
</tr>
<tr>
<td></td>
<td><code># result is a list of SpellCheckerResult objects.</code></td>
</tr>
</tbody>
</table>
5. Web Interface

Tiny MCE web editor is used as framework to enable interactive editing and spellcheck function from this editor is integrated into the server running tamilinayavaani in backend as a Flask application.

Fig. 2.a. Invoking the spell checker running in the Flask application server in backend.

Fig. 2.b. Erroneous words indicated in the text/web-interface.
Fig. 1.c. Alternatives for first erroneous word

Fig. 2.d. Alternatives for second word after correcting first word.
Further work can involve providing a common API for all Tamil spelling checkers. A good (common) interface would be pyspeller project including the following methods:

1. API to check if its known or unknown (Dictionary membership)
2. Set edit-distance=2 (default)
3. Correction: method to yield best alternate suggestion to word
4. Suggestions: alternates for misspelled word.

6. Limitations

Even though the code for Desktop Vaani is made open-source by TVA, the file format and design of the JSON file is kept proprietary making it hard to update the word-list and/or rules associated with the spelling error correction. This is somewhat of a major limitation to update the spell checker compared to the other alternatives like Hunspell, Aspell, etc. mentioned in sec.1 which allow generation of custom affix files for expanding vocabulary of the spell checker. This limitation stands despite the ability of the spell checker to support the custom user word list, as is standard in practice.

7. Improving performance with golden words corpus

We have collected 25,83,000 unique tamil words and 1,53,548 unique tamil nouns from various public available tamil text and released as public repositories. We are cleaning these strings to build a golden words of corpus. We can use these words for quick lookup for a given word is there in the corpus or not. If not, we can then apply the invoke the tamilinayavaani spellcheker. This will help to increase the performace on processing large amount of text.

8. Conclusion

In this work we report successful creation of an open-source component spelling checker, named tamilinayavaani, using the C#(C-sharp) open-source desktop version of Vaani, from TVA. We updated the base version of the software to use sandhi checker from work of Nithya [3]. We integrated this open-source component into a web based user interface via Tiny MCE, and published a Python package for use by everyone under terms of same GPLv2 license.
References

1. Tamil Virtual Academy publication of Desktop Vaani
11. All Tamil Words - https://github.com/KaniyamFoundation/all_tamil_words
12. All Tamil Nouns - https://github.com/KaniyamFoundation/all_tamil_nouns
அல்லது 2020

வளைவம் 11, 12 & 13.

கலைவக குறுகிய கூட்டுச்சேரும் மாணாணரின் பண்பாட்டு

புனிதமான மனிதர்கள் நிர்வாகமானம்

அதிமித்யாதிகாரம் பரணாளிகளுக்கு கவண்டு

சுத்தமான குறிப்பிட்டு, சுமாரம் திருமலினாய் பல்கலைக் கழகம்

தினமண்டவார்

கலைவக குறுகிய கூட்டுச்சேரும் மாணாணரின் பண்பாட்டில் நேர்நோக்கிய ஆடை உள்ளது. அவியலம் தொழில்நுட்பம் மாணாணரின் வந்த கருத்துக்கான ரீதியாகவும், புனிதமான மனிதர்களை நோக்கிய தொடர்வகுவாயிலிருந்து. மேலும் தொழில்நுட்பம் நிறுவன அல்மத்தகக் கூட்டுச்சேரும் நூற்றாண்டுகளுக்கு முந்தைய கலை வளைவக கழகம் வழியாக, கணினிப் பொறியியல் பணியூட்டுபவர் குறுகிய கூட்டுச்சேரும் மாணாணரின் பண்பாட்டில் குறிப்பிட்டு பல்கலைக் கழகமால் குறிப்பிட்டு.

அளிக்கவும் மாணாணம் புள்ளியியல் 'ஒகாணி-19' என்னும் புள்ளியியல் வரவு கணினிப் பொறியியல் பல்கலைக் கழகமால். இது இவாணையில் வாங்க வந்த அதி மிதமான பொழுதுபோக்கினை 'Home based Learning' என்றுக் குறிப்பிட்டு, 'கலைவக குறுகிய கூட்டுச்சேரும்' (Blended Learning) அறிவிய கூட்டுச்சேரும் வழியாகப் பல்கலைக் கழகம் பல்கலைக் கழகம் பல்கலைக் கழகமால் அடிக்கடி வந்த குறிப்பிட்டு.

'ஒகாணி-19' எனும் பல்கலைக் கழகம்

இது கூட்டுச்சேரும் மாணாணம் கூட்டுச்சேரும் வரவு கூட்டுச்சேரும் 'ஒகாணி-19' என்னும் புள்ளியியல் வரவு கூட்டுச்சேரும். இதனால், மாணாணர்கள் தொழில்நுட்பம் கூட்டுச்சேரும் தொழில்நுட்பம் பல்கலைக் கழகமால். மாணாணராலும் சுத்தமான குறிப்பிட்டு நூற்றாண்டுகளின் நூற்றாண்டுகள் முழுமையான குறிப்பிட்டு. இது புள்ளியியல் வரவு கூட்டுச்சேரும் விளையாட்டுக்கு மாற்றும் குறிப்பிட்டு. அது வரவு 'Home based Learning' (HBL) பல்கலைக் கழகம். இந்த பல்கலைக் கழகத்தின் அளிக்கும் புள்ளியியல் கூட்டுச்சேரும் தொழில்நுட்பம் கூட்டுச்சேரும் தொடர்வகுவாயிலிருந்து. அது என்பது 'இந்த பல்கலைக் கழகம்' (HBL) பல்கலைக் கழகம். இந்த பல்கலைக் கழகத்தின் அளிக்கும் புள்ளியியல் கூட்டுச்சேரும் கூட்டுச்சேரும் தொடர்வகுவாயிலிருந்து. 'ஒகாணி-19' வரவு கூட்டுச்சேரும் மாணாணம் மாணாணராலும் சுத்தமான குறிப்பிட்டு வந்த குறிப்பிட்டு குறுகிய கூட்டுச்சேரும் தொடர்வகுவாயிலிருந்து. மாணாணரின் பண்பாட்டில் நேர்நோக்கிய ஆடை உள்ளது. அவியலம் தொழில்நுட்பம் மாணாணரின் வந்த கருத்துக்கான ரீதியாகவும், புனிதமான மனிதர்களை நோக்கிய தொடர்வகுவாயிலிருந்து. மேலும் தொழில்நுட்பம் நிறுவன அல்மத்தகக் கூட்டுச்சேரும் நூற்றாண்டுகளுக்கு முந்தைய கலை வளைவக கழகம் வழியாக, கணினிப் பொறியியல் பல்கலைக் கழகமால் குறிப்பிட்டு.

143
புளியநாயகி குரு தினசரி மார்ச்சு, 2020

பதாக்கத்தில் அறிக்கையிடி தற்போது மதியான் குறிப்பிட்டுள்ள பட்டைப்பொறிகளுக்குப் பதினும் திருவுருவத்திலிருந்து போய்வானது, மேலும் பாதுகாப்பான துவாரமுறை குறிப்பிட்டு திருளித்துள்ள அவர்களுடன் தாங்கியது, அவர்களிடமிட்டு இவ்விதமான மதியான அவர்களிடமிட்டு துவாரமுறை குறிப்பிட்டு மதியான மாமலை அவர்களுக்கு.

17 https://www.nie.edu.sg/teacher-education/useful-resources-for-HBL-WFH-EdCo/tips-for-faculty-teachers-for-home-based-learning(hbl)

144
புதுவைக்குடும்பத்தில் நடந்த நூற்றாண்டுப் போட்டியில், 2020
புவரம் 11, 12 & 13.

சதுரத்து பள்ளைக் கல்வியல் பயிற்சியைத் தயாரிப்பு வருமாறுகோள் மேலாண்டு பயிற்சியைத் தயாரிப்பு மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறுகோள். உடையவர், தேவாரம்பியம் எனும் வங்கிய மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறு குறிப்பிட்டு வருமாறு. உடையவர், மேலாண்டு தூய்தண்டிற்கும், தேவாரம்பியம் எனும் வங்கிய மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறு குறிப்பிட்டு வருமாறு. உடையவர், தேவாரம்பியம் எனும் வங்கிய மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறு குறிப்பிட்டு வருமாறு. உடையவர், தேவாரம்பியம் எனும் வங்கிய மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறு குறிப்பிட்டு வருமாறு. உடையவர், தேவாரம்பியம் எனும் வங்கிய மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறு குறிப்பிட்டு வருமாறு. உடையவர், தேவாரம்பியம் எனும் வங்கிய மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறு குறிப்பிட்டு வருமாறு. உடையவர், தேவாரம்பியம் எனும் வங்கிய மேலாண்டு பயிற்சியைத் தயாரிப்பு வருமாறு குறிப்பிட்டு வருமாறு.

கவனாமன்கு கச்சிக் குறிப்பிட்டு அந்தா பண்டிகை புனிதநாயகச் சூரியன் அம்மாய். அது ஆர் பயிற்சியை புனிதநாயகச் சூரியன் அம்மாய். அது பாஸ்கர் பாலாஜீ சூரியன் அம்மாய். 

145
புதுக்கோட்டையூர் மின் திகையம் மங்கள், 2020
புதுவை 11, 12 & 13.

மிதியா மின்கையல் வழியாக ஓரையையே தரும் புதுக்கோட்டையூர் மின்கையல் மங்கள்.

கலைவைற கிளையும் பயிற்சிப்படுத்துவதற்கான தமிழ் இணைய மாநாடு, 2020

11, 12 & 13.

ச.பக்தாவாச விளக்கத்திற்கான தமிழ் போராட்டங்களை பிரபலமாகக் கொள்ள வேண்டும்.

146

20 https://iteachwell.blogspot.com/2020/?m=0
21 https://iteachwell.blogspot.com/2020/?m=0_Mobile learning
பொதுகாலக்கணிப்புக் கல்வி திறன் வருவத், 2020
பகுதி 11, 12 & 13.

கல்விக்கணிப்புக் குழுவால். அந்தக் குழுவின் செயல்பாடுகளுக்கு தரமான குழுவுக்கும் அரசுத் தொடர்புகாண்பாடுகளும், விளையாட்டும், பணிமன்றங்கும் வரபுருஷாக செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் குழுப்பாதாரம் நிறுவும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும்.

பாடல்களில் கல்விக்கணிப்புக் குழுவால் கல்விக்கணிப்புக் குழுவின் செயல்பாடுகளுக்கு தரமான குழுவுக்கும் அரசுத் தொடர்புகாண்பாடுகளும், விளையாட்டும், பணிமன்றங்கும் வரபுருஷாக செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் குழுப்பாதாரம் நிறுவும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும்.

அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும். அந்தக் குழுவின் பணிமன்றங்களுக்கு செயல்பாடுகளுக்கு கல்விக்கணிப்புக் குழுவின் பணிமன்றங்களுக்கு நாச்சுவை வழங்கும்.
புதினைச்சுருக்காத் துற்ற திளக்கு மாடல், 2020
பாதுகாம்ப 11, 12 & 13.

மாணவர்கள் நிரும்போம்பங்கள் பல்கலைக்கழகத்தில். மாணவர்கள்
கேரளத்தில் இயற்றிய பாதுகாம்ப ஆண்டு குறிப்பிட்டியதாக.

இந்திய ஆர்வத்தம்மான

மாணவர்கள் ஆர்வத்தம்மான தமிழகத்தில் காணப் பெற்றவர்கள். அதன்போது கால்
மாணவர்கள், தமிழ், சமயம், அனைத்து மொழிகளில் காணப்
பயன்படுத்தப்பட்டுள்ள கருத்துக்கான வாசிக்கலர் முன்னேற்றிய ஆர்வத்தம்மான
பாதுகாம்ப தொடர் விளக்கம். 'கேரளாவில்-19' ரோஸ்ட்ரம்செயல்களில் அறிமுகப்படுத்தப்பட்ட செயல்களின்
குறிப்பிட்டல். இந்திய ஆர்வத்தம்மான மாணவர்கள் ஆர்வத்தம்மான
செயல்களில் முன்னேற்றிய ஆர்வத்தம்மான குறிப்பிட்டியதாக.

குறிப்பிட்டிய முறைப்படுத்தும் ஆண்டு.

குறிப்பிட்டிய முறைப்படுத்தும் ஆண்டு பல்வேறு சிற்றுயிரியாரியால் ஆண்டுக்கு குறிப்பிட்டியதாக
முன்னேற்றப்பட்டுள்ள பல்கலைக்கழகம் குறிப்பிட்டியதாக. அதுயனவியார்கள் குறிப்பிட்டியதாக
'சிதை விதைக் குறிப்பிட்டியதாக. இந்திய ஆர்வத்தம்மான
முன்னேற்றிய பல்கலைக்கழகம் குறிப்பிட்டியதாக. தமிழக ஆர்வத்தம்மான
செயல்களில் முன்னேற்றிய ஆர்வத்தம்மான குறிப்பிட்டியதாக.
பணிக்கூட்டம் பதிப்பு  திசையன் வருடம், 2020

பதிப்பு 11, 12 & 13.

உண்மையான பிரதானச் சேவைகள் மற்றும் கல்வியான கட்டுமான வளாகத்தில் முக்கியத்துவமைத்துக். இதனால் அந்தக் கோபத்தை தொடர்பு அடர்த்திக்கப்பட்டு ஒரு அருகில் உள்ளராக அதிகம் செய்யப்பட்டது. அந்தந்த உள்ளராக வேளாண்மை காணப்பட்டது.

என்னையுடன் என
பதிகச் சாதனைகள்  நவம்பர் 2020
9 வருடம் 11, 12 & 13.

மாதிரியான பார்வாலித்துண்டு

நூற்றண்ணாவது பாண்டியரின், புனித ஆடலமாணவன்

பதின் பிரிவு அழைக்கு: ezhillang@gmail.com

1330 அல்லது பார்வாலித்துண்டு மாதிரியான பார்வாலித்துண்டு விளையாட்டை அமைத்து ஓர் நோக்கியுள்ளது? நூற்றண்ணாவது பாண்டியர் தேவாரபியர் பார்வாலித்துண்டு ஓர் நோக்கியுள்ளது? புனித புனித புனித ஓர் நோக்கியுள்ளது?

ஓமாமதே - அரசியல்

அரசியல் பாண்டியர் அவர் சந்திரத்தில் தனது அளவுகளில் பெரித்துனர் நிறுவனங்கள் தொகுதி பாண்டியர் அளவியற் வழி முறையே முறையே ஆரம்பம் வருகிறது. இது புனித புனித புனித புனித புனித.

புனித புனித புனித புனித புனித புனித புனித புனித புனித புனித

தேவாரபியர் 'kural.Thirukkural().get_kural_no()' என்ற பகுதியைத் தயாரித்து மற்றும் மற்றும் மற்றும் மற்றும் மற்றும் மற்றும் மற்றும் மற்றும் 'kural.Thirukkural().get_kural_no()' என்ற பகுதியைத் தயாரித்து மற்றும் மற்றும் மற்றும் 'kural.mathirai.py' என்ற பகுதியைத் தயாரித்து மற்றும் மற்றும் மற்றும் மற்றும் மற்றும் 'kural.mathirai.py' என்ற பகுதியைத் தயாரித்து மற்றும் மற்றும் மற்றும் மற்றும் மற்றும் மற்றும் 'kural.mathirai.py' என்ற பகுதியைத் தயாரித்து மற்றும் 'kural.mathirai.py' என்ற பகுதியைத் தயாரித்து 'kural.mathirai.py' என்ற பகுதியைத் தயாரித்து

150
புதியாக்கூறுக்குரிய திசையண மாற்றம், 2020
பக்கங்கள் 11, 12 & 13.

பக்க 1: கிறித்தீன் புதியாக்கூறுக்குரிய திசையண முறையில் குறுகான்முறையில் பிரமாண்டமாக அகற்றப்பட்டது.

முறைகள்

1. கிறித்தீன் ஏற்றமாக்கல் திசையண மாற்றம் திசையண அளவு µ ~ 29.5. காரணமாக மாற்றமலை σ ~ 2.5
2. மாற்றங்கள் பாசையான கிறித்தீன் ஆற்றமாக்கல் திசையணமும் பிரமாண்டமும் அம்மற்றையே உள்ளது
3. கிறித்தீன் பாசையான முறையில் குறுகான்முறையில் குறுகான்முறையின் முறையில் மற்றும் பிரமாண்டமும் உள்ளது
4. தகாலும் ஆற்றமாக்கல் திசையண (23) காரணமாக காமாநிலன, 391, 426, 483, 786

"கூறுக்குரிய திசையண மாற்றங்கள்

கூறு அதிகமாக என்பது" கூறு 391.

5. ஆற்றங்கள் பிரமாண்டமாக்கல் (37.5) காரணமாக காரணமாக காமாநிலன,

"காமாநிலன் காரணமாக காரணமாக்கல் காமாநிலன்

காமாநிலன் காரணமாக்கல் மற்றும்." கூறு 849.

கிறித்தீன் பாசையான பிரமாண்ட திசையண சான்றியியல்

<table>
<thead>
<tr>
<th>திசையண ஆளவு</th>
<th>காண்டர் தோட்டத்தியை காணும் எண்</th>
<th>காண்டர் தோட்டம்(கள்)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>4</td>
<td>234391, 426, 483, 786</td>
</tr>
<tr>
<td>23.5</td>
<td>5</td>
<td>67, 77, 366, 637, 979</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>108, 485, 961, 965, 1042, 1048, 1277, 1304</td>
</tr>
<tr>
<td>24.5</td>
<td>7</td>
<td>304, 467, 602, 652, 1118, 1322, 1324</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>133, 193, 331,360, 412,477, 546, 559, 576, 592, 616, 771, 796, 846, 947,1052, 1116, 1239, 1240, 1289</td>
</tr>
<tr>
<td>25.5</td>
<td>29</td>
<td>96, 339, 347, 350, 373, 400, 405, 439, 444, 569,</td>
</tr>
<tr>
<td>Volume</td>
<td>Number</td>
<td>Pages</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>26</td>
<td>37</td>
<td>[1, 28, 40, 90, 152, 229, 308, 314, 340, 454, 466, 480, 488, 594, 622, 639, 666, 668, 687, 692, 715, 774, 824, 847, 856, 934, 964, 975, 988, 1008, 1028, 1168, 1238, 1242, 1256, 1279, 1309]</td>
</tr>
<tr>
<td>26.5</td>
<td>46</td>
<td>[45, 54, 107, 168, 175, 196, 236, 296, 364, 451, 484, 516, 518, 531, 540, 571, 572, 598, 608, 611, 623, 628, 700, 706, 708, 738, 769, 838, 854, 877, 935, 954, 1010, 1066, 1071, 1131, 1132, 1134, 1222, 1224, 1227, 1229, 1232, 1233, 1296, 1302]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Temperature (°C)</td>
<td>Frequency (Number of Occurrences)</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>32.5</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>33.5</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>34.5</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>35.5</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>36.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>37.5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>38.0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>39.5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- The temperature range is from 32.5 to 39.5 °C.
- The frequency indicates the number of occurrences of each temperature range.
#!/usr/bin/env python3
# This Python file uses the following encoding: utf-8
from kural import Thirukkural
from tamil.utf8 import get_letters, get_tamil_words, total_maaththirai
from collections import Counter, OrderedDict
from pprint import pprint
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
from scipy.optimize import curve_fit

# Define model function to be used to fit to the data above:

def gauss(x, *p):
    A, mu, sigma = p
    return A*np.exp(-(x-mu)**2/(2.*sigma**2))

def main():
    eq = Counter()
    eqd = {}
    kural = Thirukkural()
    for kural_no in range(1330):
        kural_words = get_tamil_words(get_letters(kural.get_kural_no(kural_no+1).ta))
        mathirai = sum([total_maaththirai(word) for word in kural_words])
        if eq[mathirai] == 0:
            eqd[mathirai] = [kural_no+1]
        else:
            eqd[mathirai].append(kural_no+1)
        eq[mathirai] += 1
    eq_sorted=OrderedDict(sorted(eq.items(),key=lambda x: x))
    print("total = ",sum(eq.values()))
    plt.scatter(eq_sorted.keys(),eq_sorted.values())
    plt.ylabel(u'எணி%ைக்', {'fontname': 'Catamaran'})
    plt.xlabel(u'மாதிரிைர் அளிக்கும்', {'fontname': 'Catamaran'})
    # p0 is the initial guess for the fitting coefficients (A, mu and sigma above)
    p0 = [75., 20., 5.]
    coeff, var_matrix = curve_fit(gauss, list(eq_sorted.keys()), list(eq_sorted.values()), p0=p0)
    hist_fit = gauss(list(eq_sorted.keys()), *coeff)
    plt.plot(eq_sorted.keys(), hist_fit, label='Gaussian Fitted data (mean=%g, std=%g)'%(coeff[1],coeff[2]))

if __name__ == '__main__':
    main()
முன்னாள் கல்விக் குழுவான கல்விக் குழுவான, 2020
போட்டி 11, 12 & 13.

முன்னாள்

1. முன்னாள், முன்னாள் முன்னாள்.

2. குழுவான குழுவான குழுவான முன்னாள் வரலாறு: 0.97, (முன்னாள், 2020).
https://pypi.org/project/Open-Tamil/
Generation and Parsing of Number to Words in Tamil

Authors: Muthiah Annamalai, Sathia Mahadevan

Abstract
We discuss generation and parsing of Tamil words from and to numbers respectively. We show this task can be achieved in O(n) complexity and performs well for written text. We explore features of these algorithms within a voice user-interface based calculator; we also list some of the limitations. These algorithms are implemented in the Open-Tamil library.

1. Introduction
Automatic evaluation of mathematical expressions in the context of natural-language processing (NLP) has been carried out in English language as far back as the 1960s [1]. Voice interfaces to various banking, token based queue systems in retail and wholesale spaces, are desirable to read-out numbers as words in the specific language; conversely systems interacting by speech input are also desired to take number input as spoken words in the specific language. To this effect we need systems to generate the numbers to words and parse text back to numbers.

Not much work is reported on number to word generation in Indian languages in general and Tamil in particular, to best of our knowledge. Tamil numbers have specific descriptions in written and spoken forms of language [2], [6], [7].

2. Generation
Number forms in Tamil have recursive properties and details of their nuances in dealing with 90s, 900s, 9000s numbers and 10-19 numbers are elaborated in standard reference books like [2], [6], [7].

Typically, multiplication tables (வாடுலடு) are used in grade school (K-12) to commit basic mathematical facts to memory and take a textual form as “இரநாபா”, meaning 2 x 4 = 8. Ability to generate numbers in word-forms enables us to generate these multiplication tables ad-hoc. [4], [5].
2.1. Algorithm
Algorithm works for generating integral and floating point non-negative numbers; it generates text forms of numbers in both Indian (i.e. using lakhs, crores) and American standard (i.e. using millions, billions). Lakh is 100,000, Crore is 100 Lakhs, and Million is 10 Lakhs. We present the algorithm for Indian standard number to word conversion using base-10:

Input: floating point number \( N \)
Output: string of Tamil words \( T \)

1. Load list of prefix and string suffix for all Tamil number words - 63 words in all.
2. Find the quotient \( Q \), remainder \( R \) for \( N \) divided by 1 crore, lakh, thousand, hundreds, or tens
3. If \( Q \) is zero set \( N=R \) and continue to 2.
4. Convert the quotient to words \( T \)
   a. Take special care to handle 90s, 900s, 9000s, correctly.
   b. Take special care to handle number in 11-19.
5. Invoke same algorithm recursively for remainder \( R \).
6. Concatenate results from 5 to $T$
7. Return $T$

Similarly we get the American standard number to word conversion if we replace step 2 in above algorithm to divide by trillion, billion and million in replacement of crore, lakh placeholder values.

The code for this algorithm is shown in subroutine `num2tamilstr` in Appendix A.1. The code for American standard number to word generation code is similar but not shown in this paper; one can refer to [4b].

2.2. Performance
The current algorithm is $O(n)$ for $n$ significant digits of the input specified as floating-point number. The decimal point, pulli, is handled correctly and the current algorithm generates a much larger verbosity of full-precision floating point input upto the number of significant digits present in the computer memory representation of double.

3. Parsing

3.1 Algorithms
Algorithm works for parsing integral and floating point non-negative numbers; it parses text forms of numbers in both Indian (i.e. using lakhs, crores) and American standard (i.e. using millions, billions). It is roughly the converse of the algorithm in section 2.

**Input**: string of Tamil list of words $T$
**Output**: floating point number $N$

1. Load list of prefix and string suffix for all Tamil number words - 63 words in all.
2. Initialize $N$ at 0
3. Create temporary stack $S$
4. FOR word $W$ in $T$
5. IF $W$ in stop words (crores, lakhs, thousands, hundreds, tens)
   a. Convert words in stack $S$ into value and scale temporary result $N$ using a helper routine which handles input upto value 100,000.
   b. Empty stack $S$
6. ELSE: push $W$ into $S$
7. END loop started at 4.
8. Stack S is mostly non-empty and you have to use a helper routine to get the final portion of the number using the same helper function in 5a.
9. Correctly parsed value is stored in $N$

Helper routine also understands decimal point (pulli) and returns a single number which forms a scale for the intermediate multiple of 10. The code for the algorithm and helper algorithm are shown in subroutines `tamilstr2num` and `helper_tamilstr2num` respectively in Appendix. A.

### 3.2 Performance

This algorithm is linear in complexity $O(n)$ for the number of words of text in the input. The accuracy of the parsed number depends on the floating point representation of the hardware - and code uses 64-bit double representation as implemented now.

### 4. Results and Applications

![Fig. 2. Voice User-Interface for Tamil Spoken Calculator.](image)

The concept of a spoken voice user-interface (VUI) calculator is shown in Fig. 2. While such a system has been proposed for English and other languages, e.g. in patent applications [3a,b] etc. our work forms the foundation of such a system in Tamil. Key limitations of [3] are the requirement of a generator/parser for numerals in the target language as well as the speech recognition and synthesis facilities.

Using or extending this technique to associating numerals, ordinals, and number conjugates [7] to nouns in a corpus will help improve the resolution and fine-tune named entity recognizers, by adding these quantitative attributes on such named-entities.
Table 1. Expression Evaluation

<table>
<thead>
<tr>
<th>Input Expression</th>
<th>Parsed Expression</th>
<th>Evaluated Result</th>
<th>Output Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ஒன்று கால் ஒன்று</td>
<td>1 + 1</td>
<td>2</td>
<td>இரண்டு</td>
</tr>
<tr>
<td>ஒன்று கால் மூன்று பால்கள்</td>
<td>1 + 2 * 10</td>
<td>21</td>
<td>இருபது</td>
</tr>
<tr>
<td>அந்த புனிதம் உள்ளிட்டு புகழ்மையுடைய ( ஒன்று கால் ஒன்று )</td>
<td>1000 - 5*(9 + 1)</td>
<td>950</td>
<td>ஐந்து ஐந்து பால்கள்</td>
</tr>
<tr>
<td>பதினாயிரம் பால்கள் பால்கள்</td>
<td>100000*10</td>
<td>1000000</td>
<td>பதினாயிரம்</td>
</tr>
</tbody>
</table>

4.1. Results
To this extent we show core concepts (orange, green coded blocks) of such spoken mathematical expressions in Tamil and their evaluation with Open-Tamil library application named olini (ோலிலி) [4]. This expression evaluator understands infix expressions including binary operators of addition, subtraction, multiplication and division including parentheses. While a full integration of the speech recognizer (ASR) and text-to-speech (TTS) modules have not been accomplished, the resulting performance can be summarized in Table. 1.

4.2. Limitations
Current work is limited to perfectly spelled text form of numbers, and does not work for text input in spoken Tamil dialect, or text with spelling errors. However, there are simple remedies to these contexts.

Future work, could include parsing of text with morphological analyzer and tolerance toward spelling errors in text by using spelling corrector. Further work, can also involve moving towards generation/recognition of the various forms of Tamil number words - fractions (அஷைய,காணை, காட்கள், ... பால் பல்), ordinals, multiplication tables (தமையகமாறி), domain specific uses in colloquial spoken contexts e.g. "இந்தோலைம் பால்கள் காலு", "பால் தைசுக்கையாய்," [2],[7] etc.

We hypothesize the algorithm for generating number to words is identical in other Dravidian languages due to the underlying relationships of the languages. We suspect this may also hold true for TTS generation of audio of the Numerals. This remains to be seen.
5. Conclusion
This work presents, for the first time to our knowledge, a comprehensive account of generating and parsing Tamil number words. We show applications of our work to a future voice user-interface based spoken calculator, and even perhaps name-entity recognizers.

6. References
A. Appendix - Python Source Code

A.1 Generating Tamil Number Words

```python
def num2tamilstr( args):
    """ work till one lakh crore - i.e 1e5*1e7 = 1e12.
    turn number into a numeral, Indian style. Fractions upto 1e-30""
    number = args[0]
    if len(args) < 2:
        filenames = []
    else:
        filenames = args[1]
    if len(args) == 3:
        tensSpecial = args[2]
    else:
        tensSpecial = 'BASIC'
    if not any(filter(lambda T: isinstance(number, T), [str, int, float])) or
       not isinstance(number, complex):
        raise Exception("num2tamilstr input has to be an integer or float")
    if float(number) > int(1e12):
        raise Exception("num2tamilstr input is too large")
    if float(number) < 0:
        return u"" + num2tamilstr(-number)

    units = (u'ஒன்று', u'ஒன்று', u'ொன்று', u'ஒன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று')  # 0-10
    units_suffix = (u'ஒன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று')  # 0-10
    units_suffix_nine = (u'ஒன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று')  # 0-10
    tween = [1, 2, 5, 0, 6, 0, 7, 0, 8, 0, 9, 0]
    teens = (u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்')  # 11-19
    tens = (u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்', u'ைதிழானால்')  # 10-90
    tens_full_prefix = (u'ஒன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று')  # 10+-90+
    tens_prefix = (u'ஒன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று', u'ொன்று')  # 10+-90+
```
```
hundreds = ( '상', '십', '백', '천', '천', '천',
             '천', '천', '천', '천', '천', '천') #100 - 900
hundreds_suffix = ( '십', '십', '백', '천', '천', '천',
                    '천', '천', '천', '천', '천', '천')
                    #100+ - 900+

one_thousand_prefix = '만'
thousands = ( '천', '천', '천')

one_prefix = '열'

lakh = ( '십억', '백억', '천억')
crore = ( '억', '억', '억')

pulli = '슬'
n_one = 1.0
n_ten = 10.0
n_hundred = 100.0
n_thousand = 1000.0
n_lakh = 100.0*n_thousand
n_crore = (100.0*n_lakh)

# handle fractional parts
if float(number) > 0.0 and float(number) < 1.0:
    rval = []
rval.append(pulli)
filenames.append( 'pulli' )
number_str = str(number).replace('0.',"")
for digit in number_str:
    filenames.append( "units_%d"%int(digit))
rval.append( units[int(digit)] )
return u'
'.join(rval)

if isinstance(number,str):
    result = u"
    number = number.strip()
    assert(len(args) == 1)
    assert(len(number) > 0)
    is_negative = number[0] == "-"
    if is_negative:
        number = number[1:]
frac_part = u""
if number.find(".") >= 0:
    rat_part,frac_part = number.split(".")
    frac_part = num2tamilstr(u"0."+frac_part)
else:
    rat_part = number
if len(rat_part) > 0:
    result = num2tamilstr(float(rat_part))
    result = result + u"" + frac_part
return is_negative and '-' + result.strip() or result.strip()

suffix_base = { n_crore: crore, n_lakh : lakh, n_thousand : thousands}
suffix_file_map = {n_crore: "crore", n_lakh : "lakh", n_thousand : "thousands"}

file_map = {n_crore :["one_prefix","crore_0"],
    n_lakh : ["one_prefix","lakh_0"],
    n_thousand : ["one_thousand_prefix", "thousands_0"],
    n_hundred : ["hundreds_0"], #special
    n_ten : ["units_10"],
    n_one : ["units_1"]}

num_map = {n_crore : [one_prefix,crore[0]],
    n_lakh : [one_prefix,lakh[0]],
    n_thousand : [one_thousand_prefix, thousands[0]],
    n_hundred : [hundreds[0]], #special
    n_ten : [units[10]],
    n_one : [units[1]]}

all_bases = [n_crore, n_lakh, n_thousand, n_hundred, n_ten,n_one]
allowed_bases = list(filter( lambda base: number >= base, all_bases ))
if len(allowed_bases) >= 1:
    n_base = allowed_bases[0]
    if number == n_base:
        if tensSpecial=='BASIC':
            filenames.extend(file_map[n_base])
            return u"".join(num_map[n_base])
        elif tensSpecial=='NINES':
            filenames.extend(file_map[n_base])

166
return units_suffix_nine[int(number%10)]
else:
    filenames.extend(file_map[n_base])
    return units_suffix[int(number%10)]

quotient_number = int( number/n_base )
residue_number = number - n_base*quotient_number

# print number, n_base, quotient_number, residue_number, tensSpecial
if n_base == n_one:
    if isinstance(number,float):
        int_part = int(number%10)
        frac = number - float(int_part)
        filenames.append("units_%d"%int_part)
        if abs(frac) > 1e-30:
            if tensSpecial=='BASIC':
                return units[int_part]+u' ' + num2tamilstr(frac,filenames)
            elif tensSpecial=='NINES':
                return units_suffix_nine[int_part]+u' ' + num2tamilstr(frac,filenames)
            else:
                return units_suffix[int_part]+u' ' + num2tamilstr(frac,filenames)
        else:
            if tensSpecial=='BASIC':
                return units[int_part]
            elif tensSpecial=='NINES':
                return units_suffix_nine[int_part]
            else:
                return units_suffix[int_part]
else:
    if tensSpecial=='BASIC':
        filenames.append("units_%d"%number)
        return units[number]
    elif tensSpecial=='NINES':
        filenames.append("units_%d"%number)
        return units_suffix_nine[number]
    else:
        filenames.append("units_%d"%number)
        return units_suffix[number]
elif n_base == n_ten:
    if residue_number < 1.0:
        filenames.append("tens_%d"%(quotient_number-1))
    if residue_number == 0.0:
        return tens[quotient_number-1]
# else: //seems not reachable.
#    numeral = tens[quotient_number-1]
elif number < 20:
    filenames.append("teens_%d"%(number-10))
    residue_number = math.fmod(number,1)
    teen_number = int(math.floor(number - 10))
    if residue_number > 1e-30:
        return tens[teen_number-1] + u' ' +
    num2tamilstr(residue_number,filenames)
else:
    return tens[teen_number-1]+u' '}
if residue_number < 1.0:
    filenames.append("tens_%d"%(quotient_number-1) )
    numeral = tens[quotient_number-1]+u' '
else:
    if residue_number in tween:
        filenames.append("tens_prefix_%d"%(quotient_number-2) )
        numeral = tens_prefix[quotient_number-2]
        tensSpecial='SPECIAL'
        if (quotient_number==9):
            tensSpecial = 'NINES'
    else:
        filenames.append("tens_prefix_%d"%(quotient_number-2) )
        numeral = tens_full_prefix[quotient_number-2]+u' '
elif n_base == n_hundred:
    if residue_number == 0:
        filenames.append("hundreds_%d"%(quotient_number-1))
        return hundreds[quotient_number-1]+u' '
if residue_number < 1.0:
    filenames.append("hundreds_%d"%(quotient_number-1) )
    numeral = hundreds[quotient_number-1]+u' '
else:
    filenames.append("hundreds_suffix_%d"%(quotient_number-1))
else:
    if ( quotient_number == 1 ):
        if n_base == n_thousand:
            filenames.append("one_thousand_prefix")
            numeral = one_thousand_prefix
        else:
            filenames.append("one_prefix")
            numeral = one_prefix
    else:
        numeral = num2tamilstr(quotient_number, filenames)

if n_base >= n_thousand:
    suffix = suffix_base[n_base][int(residue_number >= 1)]
    suffix_filename = "%s_%d"%(suffix_file_map[n_base],int(residue_number >= 1))
    if residue_number == 0:
        filenames.append(suffix_filename)
        return numeral + u' ' + suffix + u' '
    filenames.append(suffix_filename)
    numeral = numeral + u' ' + suffix + u' '
    residue_numeral = num2tamilstr(residue_number, filenames, tensSpecial)
    return numeral + u' '+residue_numeral

# number has to be zero
filenames.append("units_0")
return units[0]

A.2 Parsing Tamil Number Words

def tamilstr2num(tokens):
    ""
    """
    numeral parser; convert numeral to number.
    e.g. ["இமாழி" También] => 245
    """
    if isinstance(tokens,str):
        tokens = re.split(SPACE,tokens)
    is_american_str = False
    has_decimal = False
US_values = [1e12, 1e9, 1e6]
US_placements = ['பாலாமையாட்டத்', 'பாலாமையாட்டத்', 'பாலாமையாட்டத்']
IN_values = [1e7,1e7,1e5,1e5]
IN_placements = ('உருவ்', 'சிற்றுருவ்', 'சிற்றுருவ்', 'சிற்றுருவ்', 'சிற்றுருவ்',)

for tok in tokens:
    if tok in US_placements:
        is_american_str = True
    elif tok == 'புளிப்பீதி':
        has_decimal = True
    value = 0
    stack = list()
    if is_american_str:
        HIGHEST=('பாலாமையாட்டத்',)
        PLACEMENTS=US_placements
        VALUES=US_values
    else:
        HIGHEST=IN_placements[0:2]
        PLACEMENTS=IN_placements
        VALUES=IN_values

for tok in tokens:
    if tok in PLACEMENTS:
        if len(stack) == 0:
            tmpval = 1.0
            if value != 0.0 and tok in HIGHEST:
                value = value*VALUES[PLACEMENTS.index(tok)]
            continue
        else:
            tmpval = helper_tamilstr2num(stack)
            stack = list()
            value += tmpval*VALUES[PLACEMENTS.index(tok)]
        continue
    stack.append(tok)
    if len(stack) != 0:
        value += helper_tamilstr2num(stack)
return value

def helper_tamilstr2num(tokens):
val_map = {}
val_units = list(range(11))
units = ('ொன்று', 'ொன்றுக்கு', 'ொன்றுகள்', 'ொன்றுகளை', 'ொன்றுகளில்', 'ொன்றுகளைச்', 'ொன்றுகளைத்', 'ொன்றுகளைத்தில்', 'ொன்றுகளைத்தில்லை') # 0-10
units_suffix = ('ொன்று', 'ொன்றில்', 'ொன்றில்லை', 'ொன்றில்லைத்', 'ொன்றில்லைத்தில்', 'ொன்றில்லைத்தில்லை', 'ொன்றுக்கு', 'ொன்றுக்குத்', 'ொன்றுக்குத் நேர்ந்த', 'ொன்றுக்குத் நேர்ந்து' ) # 0-10
units_suffix_nine = ('ொன்றுக்குத்', 'ொன்றுக்குத் 90+ பதூர்', 'ொன்றுக்குத் 90+ பதூர் கள்', 'ொன்றுக்குத் 90+ பதூர் களை', 'ொன்றுக்குத் 90+ பதூர் களில்') # 0-10
for _u in [units,units_suffix,units_suffix_nine]:
    for k,v in zip(_u,val_units):
        val_map[k]=v

teens = ('பதூர்', 'பதூரின்', 'பதூர் கள்', 'பதூர் களை', 'பதூர் களில்', 'பதூர் களைச்', 'பதூர் களைத்', 'பதூர் களைத்தில்', 'பதூர் களைத்தில்லை') # 11-19
for k,v in zip(teens,range(11,20)):
    val_map[k]=v

tens = ('பதிெதா', 'பதிெதா கள்', 'பதிழ்வேம்', 'பதிழ்வேம் கள்', 'பதிழ்வேம் களை', 'பதிழ்வேம் களில்', 'பதிழ்வேம் களைச்', 'பதிழ்வேம் களைத்', 'பதிழ்வேம் களைத்தில்', 'பதிழ்வேம் களைத்தில்லை') # 10-90
for k,v in zip(tens,range(10,100,10)):
    val_map[k]=v

tens_full_prefix = ('பதிெதாக்கள்', 'பதிெதாக்கள் கள்', 'பதிழ்வேம்லகள்', 'பதிழ்வேம்லகள் கள்', 'பதிழ்வேம்லகள் களை', 'பதிழ்வேம்லகள் களில்', 'பதிழ்வேம்லகள் களைச்', 'பதிழ்வேம்லகள் களைத்', 'பதிழ்வேம்லகள் களைத்தில்', 'பதிழ்வேம்லகள் களைத்தில்லை') # 10+90+
for k,v in zip(tens_full_prefix,range(20,100,10)):
    val_map[k]=v

tens_prefix = ('பதிழ்வேம்', 'பதிழ்வேம் கள்', 'பதிழ்வேம் களை', 'பதிழ்வேம் களில்', 'பதிழ்வேம் களைச்', 'பதிழ்வேம் களைத்', 'பதிழ்வேம் களைத்தில்', 'பதிழ்வேம் களைத்தில்லை') # 10+90+
for k,v in zip(tens_prefix,range(20,100,10)):
    val_map[k]=v

hundreds = ('பதிழ்வேம்', 'பதிழ்வேம் கள்', 'பதிழ்வேம் களை', 'பதிழ்வேம் களில்', 'பதிழ்வேம் களைச்', 'பதிழ்வேம் களைத்', 'பதிழ்வேம் களைத்தில்', 'பதிழ்வேம் களைத்தில்லை') #100 - 900
for k,v in zip(hundreds,range(100,1000,100)):
    val_map[k]=v

hundreds_suffix = ('பதிழ்வேம்', 'பதிழ்வேம் கள்', 'பதிழ்வேம் களை', 'பதிழ்வேம் களில்', 'பதிழ்வேம் களைச்', 'பதிழ்வேம் களைத்', 'பதிழ்வேம் களைத்தில்', 'பதிழ்வேம் களைத்தில்லை') #100+ - 900+
for k,v in zip(hundreds_suffix,range(100,1000,100)):
    val_map[k]=v

one_thousand_prefix = 'ெதா'
val_map[one_thousand_prefix] = 1.0
thousands = (u'ஆயிர%', u'ஆயிர%&')
val_map[thousands[0]] = 1000.0
val_map[thousands[1]] = 1000.0
one_prefix = u'ஒ'
val_map[one_prefix] = 1.0
lakh = (u'இலச%', u'இலச%&')
val_map[lakh[0]] = 100000.0
val_map[lakh[1]] = 100000.0
crore = (u'ேகா%', u'ேகா%&')
val_map[crore[0]] = 10000000
val_map[crore[1]] = 10000000
pulli = u'ளி'
val_map[pulli] = 0.0
mil = u'மிய&
val_map[mil] = 1000000.0
bil = u'பி&
val_map[bil] = val_map[mil]*1e3
tril = u'ாி%ய'
val_map[tril] = val_map[bil]*1e3
in_fractional_portion = False
multiplier = 1.0
value = 0.0

# ["தாசறா", "தரணப்புலாத்"] => 245
n_tokens = len(tokens)
for idx,tok in enumerate(tokens):
   n_remain = n_tokens - idx - 1
   if in_fractional_portion:
      multiplier *= 0.1
      value += multiplier*val_map[tok]
      continue
   if tok in mil:
      multiplier = 1e6
      continue
   elif tok in bil:
multiplier = 1e9

continue

elif tok in tril:
    multiplier = 1e12
    continue

elif tok in lakh:
    multiplier = 1e5
    continue

elif tok in crore:
    multiplier = 1e7
    continue

elif tok in thousands:
    multiplier = 1e3
    continue

elif tok == pulli:
    if multiplier > 1.0:
        if value > 0:
            value *= multiplier
        else:
            value = multiplier
    else:
        value *= multiplier

multiplier = 1

in_fractional_portion = True

continue

for _tens in tens_prefix:
    if tok.startswith(_tens):
        tok = tok.replace(_tens,"")
        value = value*multiplier + val_map[_tens]
        multiplier=1.0

        if tok != ":
            value += val_map[tok]
            tok = 

        continue

for _hundreds in hundreds_suffix:
    if tok.startswith(_hundreds):
        tok = tok.replace(_hundreds,"")
        value = (value)*multiplier + val_map[_hundreds]
multiplier = 1.0
if tok != ":
    value += val_map[tok]
tok = "
    continue
value = value * multiplier + (tok != " and val_map[tok] or 0)
multiplier = 1.0
if multiplier > 1.0:
    if value > 0:
        value *= multiplier
    else:
        value = multiplier
elif not in_fractional_portion:
    value *= multiplier
return value
Abstract
Spell checker is an invisible but indispensable component of a very large chunk of computer users’ lives. It comes with most softwares that manipulate text like office suites that include from simple note-taking to fully versatile word processors and search engines. Spell checkers tokenize text and verify whether the text contains spelling and grammatical errors. Existing spell checkers, though work for most european languages, do not account for linguistic features of the Indian languages like Tamil. In addition, the issues with encoding representation of Tamil under unicode adds even more complexity on how the Tamil text should be handled for applications like spell checking. We implement and test three different spell checkers for Tamil namely bloom-filter, symspell, LSTM based spell checkers. Symspell is very fast for validation and lookup suggestions. LSTM implementation though not accurate enough for day to day use, is an interesting line of work that remains unexplored.

Introduction
Tamil is one of the ancient and classical languages of the Dravidian family which dates back to 3rd century B.C. Tamil is a very rich and complex language in terms of literature, dialects, rich set of graphemes and vocabulary enabled by linguistic features like agglutinative morphology and grammar. This richness comes with its costs. Representation of Tamil script in computers is riddled with issues for instance unicode encoding of Tamil does not correspond to its natural script and how it composes uyir-meи characters. This adds even more complexity on how the Tamil text should be handled for applications like spell checking.

Spelling and grammar checking is a significant component of most web applications and search engines. Spell checking is a subdomain of natural language processing (NLP). The definition of spell checking varies widely ranging from trivial token correction implemented by simple lookup tables to sophisticated analyses including formal/informal tone detection, sentiment analysis, suggesting semantically similar alternatives, paraphrase detection for plagiarism checking, all of them require intelligent algorithms.
Spell-checker predicts the word user intended to write down by using a combination of clever data structures and algorithms in conjunction with established dictionaries, for finding the correct spelling and suggesting similar words based on distance metrics such as Levenshtein edit distance.

There exist a wide variety of spell checkers such as ispell, aspell, hunspell[1] which work well for English and other European languages. Though there are plugins to support east asian languages like Tamil they do not capture the linguistic features of the language, which greatly influences the different types of errors. In this work, we take inspiration from spell checker implementations for European languages and implement three methods for Tamil. We describe the common source of errors for both English and Tamil, to contrast and illustrate the influence of intrinsic nature underneath each language.

Ideally a spell-checker should let the user write down thoughts and ideas without the constant cognitive load of verifying every word and every sentence. The tool must be non intrusive such that the process of translating thoughts into its textual form is not hindered by spelling and grammatical mistakes very much similar to how a paper notebook maintains its silence throughout.

Sources of Errors

Common types of mistakes include **swapped characters**: sawp → swap; **character case**: LOWER → lower; **double occurrence**: twwo → two, thethe → the; **missing characters**: mising → missing. There are many mechanisms from which the errors manifest, the following list is not exhaustive.

Language Independent Errors

**Keyboard layouts**: Muscle memory developed on a particular keyboard layout will hinder speed when switching to a new keyboard layout. In the case of multilingual documents, this manifests very vividly due to frequent switching between two or more input methods.

**Homophones**: Similar sounding alphabets are used. தலையிட்டு → தலையிட்டு, செவ்வாய்ச்சு → செவ்வாய்ச்சு, ஹாண்டைக் கூட்டு → ஹாண்டைக் கூட்டு, காலம்பிரியா → காலம்பிரியா
Language Dependent Errors

*Input engine issues:* some input systems follow the scheme: \( uyir + pulli \rightarrow mei \) and some follow scheme: \( mei + uyir \rightarrow uyirmei \)

Apart from mechanistic and phonetics sources of errors above, the grammatical features of the language can give rise to a suite of errors in writing. The fundamental units of languages and its text differ greatly and are influenced by cultural history.

Even the very fundamental unit of a language, the alphabet performs different functions in English and Tamil. Graphemes in English and the alphabet in Tamil are not exactly the same. The definition and expression of the unit *word* also differs wildly. Words in English are mostly delimited both visually and grammatically whitespace except in a few special cases for instance, *New York* is considered a single word.

Tamil is an agglutinative language in which the words can be formed by attaching suffixes to the root word, e.g: காைல > காைலயி' > காைலயி'() > காைலயி'()ேத > காைலயி'()ேதயவ-

Agglutination have brings in alphabets that are not in the combined words which are a common source of error in writing Tamil, e.g: கிராம&'ற#களி' > கிராம&'ற)களி+,-.

Some are not strict grammatical rules but guidelines to elucidate a cleaner context e.g: கிராம&'ற#களி' > கிராம&'ற)களி+,-.

These different features of language and writing tools creates an exponentially large space of errors rendering spell checking a non-trivial problem. The following section describes the methods we implemented and tested.

**Methods**

In this work we implemented three different methods to realize spell checking for Tamil. The first two methods bloom-filter and symspell considers space delimited consecutive stream of alphabets as the single word unit. The following sub-section describes the dictionary corpus that forms the backbone of these two methods. The LSTM[2] based implementation however employs wordpiece[3] based tokenization using Byte Pair Encoding(BPE). This is to avoid the vocabulary explosion problem created by the agglutinative nature of Tamil. We chose BPE because of its versatility to adapt to the corpus instead of relying heavily on strict rules. On a character level, we do not directly
use unicode code points but instead use individual alphabets that respect Tamil script. It is important to note here that there are other tokenization methods that more grounded on grammar like morphology based tokenization [4 soumya]

**Corpus of Correct Words**

We gathered collections of words from [5] and Tamil etymological dictionary[6]. The entire dictionary for the spellchecking was built by merging the collected corpora. The corpora are merged and the resulting dictionary acts as the source of truth for the bloom-filter and symspell algorithms.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_tamil_nouns/all_nouns.txt</td>
<td>181185</td>
</tr>
<tr>
<td>tharavukkanam/tamil-etymological-dict</td>
<td>119754</td>
</tr>
<tr>
<td>merged</td>
<td>249056</td>
</tr>
</tbody>
</table>

![Fig-1: Trie built for the Tamil words: அசர், அசின், குரு, மணி, கல உருவமான, மாநாடு, பொண்ட, புருஷர், புலக், கலாச்சார், கூரிய. Notice that the மாநாடு in கல உருவமான is completely different from standalone மாநாடு](image-url)
Bloom Filter

Very minimal function of a spell checker is to flag from a stream of words which ones are incorrectly spelled. A simple lookup for the presence of the word in the dictionary is sufficient. But as the dictionary gets larger and larger we need an algorithm that can make this lookup faster. The bloom filter[7] is the simplest algorithm to implement that.

There are two issues with the bloom filter. First, the bloom filter can only check whether the word is correct or not. It does not have mechanisms to come up with a set of suggestions. Second, once a word is added to bloom-filter it cannot be removed.

Though the bloom filter can not be considered a spell checker as per widespread definition but it can very well act as a first line of validation in eliminating correctly spelled words, and let the more clever spell checking algorithms work only on incorrectly spelled words filtered out.

SymSpell

Most spell checking algorithms that employ a form of trie data structure for both validating whether a word is misspelled and also lookup suggestions, i.e closest correctly spelled words from the dictionary by a well defined metric. Usually the metric is Levenshtein distance. The trie is a tree data structure built by parsing the entire dictionary branching out as determined by the Levenshtein distance. Fig-1 shows an example of the trie built for the words, அமித, அமிதம், அமை, அமையா, அவர், அவரை, அவ்வை, அவ்வையும், அவ்வை, அவ்வை, அவ்வையும்.

Instead of looking from a dictionary of correct words to figure out the closest match for a misspelled word, SymSpell[8] algorithm flips the problem around. It builds a map of misspelled words to a probable list of correct words for up to a predetermined edit distance. This makes it extremely fast for lookup but requires more memory. In simple words, SymSpell precomputes probable misspellings of the entire dictionary beforehand to make the lookup much faster. Fig-2 shows an instance of symspell edits dictionary with edit distance = 2, built for the same set of words from the trie example as in Fig-1

LSTM

We extended the intuition gained from the symspell approach to sentence level in addition to employing a machine learning method for sequence modelling called LSTM. We built a corpus of corrupted sentences and effectively transformed the spell checking problem into a translation problem. In order to satisfy the data requirement by LSTM, we exploited the news corpus tamiltext-7M.txt[9] built for language modelling. The sentences from the tamiltext-7M.txt corpus are corrupted on alphabet level at random positions to generate sentences with errors while retaining the original sentence as the ground truth. This corpus can also be exploited for building an extensive dictionary of surface forms of tamil lexicon, though we have not used it in such a fashion. Fig-3 shows the sentence and word length distribution in the tamiltext-7M.txt corpus.
<table>
<thead>
<tr>
<th>111</th>
<th>112</th>
<th>113</th>
<th>114</th>
<th>115</th>
</tr>
</thead>
<tbody>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
<tr>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
<td>டி</td>
</tr>
</tbody>
</table>

Fig-2: Sample edit lookup table built by symspell with edit distance = 2 for 11, 12 & 13.
Fig 3: Distribution of sentence and word lengths in the corpus tamiltxt-7M.txt. (a) The space delimited sentence length (number of words in a sentence) distribution. (b) BPE tokenized sentence length distribution. (c) Word length distribution based on unicode encoding. (d) Word length distribution based on natural Tamil alphabets.

```
-- Code Listing --
BuildCorruptedSamples(alphabets, dictionary, text, aberration_count):
    tokens = split_by_whitespace(text)
    for i in range(0, length(tokens)):
        if aberration_count:
            if random_float() > 0.5:
                aberration_count = aberration_count - 1
                random_index = random_integer(0, length(tokens[i]))
                tokens[i][random_index] = random_choice(alphabets)
    return join_with_whitespace(tokens)
```

Fig 4: Pseudocode for generating parallel corpus for LSTM training

Training the character level LSTM with the parallel corpus of corrupted and correctly spelled sentences we achieve a score of 0.40 in exact match. Even though this score is
too small to be useful in practical application, this line of algorithms remain unexplored for spell checking in Tamil.

The LSTM was trained with tamiltext-7M.txt dataset with train test split ratio = 0.8 with embedding_dim = 200, hidden_dim = 50 for 1000 epochs. The optimizer used was vanilla SGD with learning_rate = 0.001 and momentum = 0.1

Discussion
We discussed the implementation of three different spell checking algorithms for Tamil language. Bloom-filter is very fast but offers limited usability. It cannot offer suggestions to the misspelled words. Symspell is also very fast at both flagging misspelled words and generating suggestions for the word under scrutiny. However it requires a large amount of memory and is not dynamic, i.e the symspell can not suggest new words that are in edit greater than what it was generated with. The memory requirement grows geometrically with edit distance. LSTM implementation is an interesting approach that at present, is not ready for real-time use. Though newer neural network architectures like transformers[10] can be used, it is still an open question whether the benefits will outweigh the computational complexity that they demand.

Runtime environment for the spell checker also varies wildly from end user devices like mobiles phones, laptops to services that reside in servers over the cloud. Even mobile devices come with a wide range of memory and cpu compute capacity. The spell checker implementations need to be stripped down or can be beefed up depending on where they are deployed. Sympspell can be tweaked to run on mobile devices with limited memory with a wide range of suggestions at the risk of slowing down other system services. LSTM based spell checking in mobile platforms is considerably slow and requires more engineering on model tweaking, but is well suited to run on cloud environments.

It is important to mention that it is useful to have the spelling and grammar checking tools explain why it came with the suggestion in grammatical terms. Existing spellcheckers do this by building a set of rules with patterns of errors and their corrections, described mainly in XML format by linguistic experts. However machine learning methods like LSTM can learn these rules implicitly but do not produce naturally interpretable output. On the other hand combining machine learning methods and rule based systems that can extract rules from what the LSTM has learned, we bypass the tedious initial bootstrapping of hand crafted rules. The machine learning system can be broken down into specific components like named entity recognizers and part of speech taggers and their outputs can be used to inform each other in an incremental fashion for practical use.
Conclusion
Spelling and grammar checking is effectively an AGI problem and so even though it is extremely complicated to create a completely automated spell checker, it is rewarding to attempt at making one. The existing spell checkers employ a suit of different heuristics to perform the function, under different environments. Tools like Grammarly work extremely well for English. Its existing infrastructure provides itself with access to a huge amount of data to learn from and improve its algorithms everyday. Though this work is merely a baby step in that direction, we look forward to developing such a platform for Tamil with the collective effort from the Tamil speaking community.

References
Abstract. In recent years, neural machine translation (NMT) has displayed substantial improvement in machine translation tasks, outperforming its predecessor statistical machine translation (SMT) for many language pairs. Currently, it has got massive attention from both academia and industry and reckoned as the promising direction for future exploration of machine translation. Even though it has produced human-like performance for many language pairs, in reference to the translations involving low-resource and morphologically rich languages, neural machine translation is yet to achieve a desirable quality. In the context of English-Tamil neural machine translation, since Tamil is a low resource and morphologically rich language, the quality of English-Tamil translation still remains low. In this article, we investigate the challenges in English-Tamil neural machine translation and present a survey of existing literature. We discuss the key techniques used by those research studies to tackle the challenges of English-Tamil neural machine translation and provide some insights into possible future research directions. We hope this article will serve as a starting point for researchers who are interested in English-Tamil neural machine translation.

Keywords: Neural Machine Translation, Low Resource, Morphologically Rich, Sub word segmentation, Transfer learning

1. Introduction

Tamil is the most widely used language under the Dravidian language family, currently being spoken by around 75 million native speakers. It is one of the official languages of the countries such as Singapore and Sri Lanka. Apart from the large number of Tamil people from Indian state Tamil Nadu and significant numbers from Sri Lanka, it is also spoken by the Tamil diaspora found in many countries around the world. Due to the huge existing demand for automatic and efficient translation of texts from English to Tamil and vice versa, English-Tamil machine translation is currently starting to get attention from the research community.

Currently, for machine translation tasks, NMT has become the go-to technique, outperforming its counterpart statistical machine translation. Unlike SMT which uses many sub components like language model and translation model, it is trained as an end-to-end single system which takes source text as the input and predicts the corresponding target text. Even though NMT has produced impressive results in high-resource data conditions [1,2,3], research studies have shown that these models are highly data-hungry and underperform SMT in low-resource settings [4,5]. Despite the poor performance of vanilla NMT for low resource languages, the huge potential it has shown in high resource conditions motivated the use of NMT for low resource languages while exploring other techniques that can improve low-resource NMT. Subsequently, Over the last few years, a lot of work has been done in this direction, and many new approaches have been suggested [6-23].

Monolingual data has been exploited to improve the performance of low resource NMT in [6-14]. Similarly, transfer learning techniques are also used where parallel data from high resource language pairs is used to pre-train the network of low-resource language pairs or jointly learn representations for both high-resource and low-resource language pairs [15-19]. There is also a massive increase in work on multilingual NMT (MNMT) systems that involve more than two languages [20-23]. When low resource languages share a parallel corpus with one or more pivot language(s), even without any direct parallel corpus between themselves, they have produced significant results. For a language pair, in a given domain, if it doesn't have enough data of that domain, out-of-domain parallel corpora and in-domain monolingual
corpora are leveraged to improve in-domain translation by a technique called domain adaptation.

In the context of English-Tamil machine translation, Tamil is a low-resource language which doesn't have enough parallel corpus for the NMT model to learn the translations. It also differs from English by its morphology and word order. These challenges make English-Tamil neural machine translation a harder problem. A considerable amount of research has been done in the arena of English-Tamil neural machine translation, addressing the above mentioned challenges. In the next section, we will introduce the linguistic characteristics of Tamil language and the challenges they might produce during the translation. Then, important research studies done with regard to English-Tamil NMT, the techniques used in them, their strengths and weaknesses are discussed. A short description of tools and resources available for building English-Tamil NMT systems is also presented. Finally, it provides some insights into possible future research directions.

2. Challenges in English-Tamil Neural Machine translation

Even though NMT has emerged as the most promising machine translation approach in recent years, it still struggles with a number of challenges. One of the key challenges is that the learning curve of NMT systems is steeper with respect to the amount of data used for training[24]. It results in poor translation quality for low-resource language pairs while better performance for high resource language pairs. Handling of out-of-vocabulary words and rare words also presents a great challenge for machine translation. Conventional neural MT models use a fixed size vocabulary, so the identity of rare words are not captured which makes their translation a difficult task. It is observed that sentences containing rare words tend to be translated much more poorly than those containing only common words[1,2]. To tackle this issue, recently various sub word segmentation techniques are proposed and have shown some improvement. Even though, NMT systems that operate at the sub-word level (e.g. with byte-pair encoding) perform better than SMT systems on extremely low frequency words, but still they struggle in translating low-frequency words belonging to highly-inflected categories (e.g. verbs)[24]. In this section, the challenges faced by English-Tamil NMT are discussed in detail.

2.1 Language Divergence

Tamil differs from English by its morphology and word order. Unlike English which is a morphologically simple language, Tamil is a morphologically rich language. English language mostly conveys the relationship between words using function words or location of the words and with less usage of morphology. But Tamil language expresses using morphological variations of words. It has a rich morphological structure and heavy usage of content words. Tamil translations of English function words do not independently exist because these words are coupled with Tamil content words and this leads to alignment problem[25]. Due to higher inflective nature, Tamil has a larger vocabulary of surface forms. Verbs are morphologically inflected due to tense and PNG (Person-Number-Gender) markers and nouns are inflected due to count and cases[25]. Each Tamil verb root is inflected into more than ten thousand surface word forms because of the agglutinative nature of Tamil language[26]. It creates data sparsity problem in English-Tamil NMT systems. If Tamil had a large amount of parallel training corpora which can cover a majority of Tamil surface forms, this problem can be partly sorted. But, since Tamil is also a low-resource language, it doesn't have the parallel corpora which contains all the Tamil surface. So, any new methods have to be used to handle all word forms with the help of limited amounts of data. Examples of Tamil word forms based on tenses are given in Table 1[Anand Kumar et al.2014].

There is also a notable difference between the syntax of English and Tamil language. English is an Indo-European language and Tamil is a Dravidian language. English has the word order of Subject-Verb-Object (SVO). It is a fixed word-order language. Tamil mostly takes the word order of Subject-Object-Verb (SOV),
but it is flexible, that is, its word order can change freely without affecting the grammatical meaning of the sentence. Fig. 1 (Anand Kumar et al. 2014) shows the word-order difference in English and Tamil sentences.

Fig. 1. Word order difference between English and Tamil

<table>
<thead>
<tr>
<th>Root</th>
<th>Tenses</th>
<th>Word Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>விளயாடு (vilayAdu)</td>
<td>Present+1S</td>
<td>விளயாடு-பிரிக்கும் vilayAdu-kinR-En</td>
</tr>
<tr>
<td></td>
<td>Present+3SN</td>
<td>விளயாடு-பிரிக்கும் vilayAdu-kinR-athu</td>
</tr>
<tr>
<td></td>
<td>Present+3PN</td>
<td>விளயாடு-பிரிக்கும் vilayAdu-kinR-ana</td>
</tr>
<tr>
<td></td>
<td>Past+1S</td>
<td>விளயாடு-பிரிக்கு vilayAd-in-En</td>
</tr>
<tr>
<td></td>
<td>Past+3SM</td>
<td>விளயாடு-பிரிக்கு vilayAd-in-An</td>
</tr>
<tr>
<td></td>
<td>Future+2S</td>
<td>விளயாடு-பிரிக்கு vilayAdu-v-Ay</td>
</tr>
<tr>
<td></td>
<td>Future+3SF</td>
<td>விளயாடு-பிரிக்கு vilayAdu-v-AL</td>
</tr>
</tbody>
</table>

Table 1. Tamil Tenses and Word Forms

2.2 Availability of English - Tamil Parallel Corpus
A well-known property of SMT and NMT systems is that increasing amounts of training data lead to better results. In SMT systems, it is previously observed that doubling the amount of training data gives a fixed increase in BLEU scores (Turcik et al., 2008; Irvine and Callison Burch, 2013). But, NMT exhibits a much steeper learning curve compared to SMT. Figure 2 shows the experiments carried out by Koehn and Rebecca Knowles 2017 to analyze the performance of NMT and SMT with the size of training corpus [24]. The starting performance of NMT is very low compared to SMT (BLEU score of 1.6 vs. 16.4) in low data setting (1/1024 of the data), then start to outperform SMT (25.7 vs. 24.7) with increase of data (1/16 of the data/24.1 million words). Later, it even beats the SMT system with a big language model (31.1 for NMT, 28.4 for SMT). The contrast between the NMT and SMT learning curves is quite striking. While NMT is able to exploit increasing amounts of training data more effectively, it is unable to get off the ground with training corpus sizes of a few million words or less [27]. When it comes to English - Tamil parallel corpus, the largest publicly available data is EnTam Dataset which has around 169k sentence pairs which is quite
low. So, there is a huge need to focus on creating parallel corpus for English-Tamil Language pairs.

Figure 2: The quality of SMT and NMT in low-resource conditions as observed by Koehn and Rebecca Knowles.2017

3. English - Tamil NMT: Research Work
In this section, we are investigating different research studies done under English-Tamil neural machine translation in the recent years. Regarding the survey scope, we only considered NMT research studies and left SMT or rule-based research works done for English-Tamil language pairs out of the scope except for brief description in the discussion and future direction. As we discussed in earlier sections, the vanilla NMT, that is, using conventional NMT architectures without using any other special techniques, has produced only low quality translations for English-Tamil neural machine translation due to the challenges they faced which are also discussed earlier on section 2. So, research studies related to English-Tamil NMT always incorporated some other mechanisms/techniques along with the vanilla NMT to mitigate those challenges. So, for the sake of simplicity, we categorize our survey based on the techniques or approaches used by those studies. We demonstrate the insight of these approaches, summarize their strengths and weaknesses and elaborate how they helped to improve English-Tamil neural machine translation. These approaches are described in the remainder of this section.

3.1 Sub Word Segmentation and Pre Trained Word Embeddings
Generally, in neural machine translation (NMT) systems, fixed-size vocabulary is used for both source and target languages because, with increase of vocabulary size, the model complexity and training time will also increase. When neural machine translation involves morphologically rich and highly inflectional languages, it is difficult to cover more words with a finite size vocabulary. This issue can be handled by segmenting words into smaller components and using those subword units instead of words as the fixed size vocabulary. Many subwording algorithms have been proposed on how to segment words into subword units. Byte Pair Encoding(BPE), SentencePieces(SP),WordPiece(WP) and Morphological Segmentation are some of the subwording techniques which have successfully applied for many language pairs in recent years. For English-Tamil NMT, research works have used techniques such as Byte Pair Encoding(BPE), morphological segmentation based sub wording and SentencePiece sub wording.

3.1.1 Morphological Segmentation
[K Hans and Milton et al 2017] have used morphological segmentation to improve the performance of English to Tamil NMT. They used morphological segmentation before word vectorization to split the morphologically rich Tamil words into their respective morphemes before the translation. It also caused a reduction in the target vocabulary size by a factor of 8. This model got 7.05 BLEU improvement over the model which didn’t use morphological segmentation (that is, word based model). Other than BLEU scores, authors compared the performance using human evaluators and the model which used morphological segmentation outperformed the word-level model.

The corpus selected for this experiment was a combination of different corpora from various domains. It comprised the EnTam v2 corpus (Ramasamy, Bojar, and Žabokrtský 2014), tourism corpus that was obtained from TDIL (Technology Development for Indian Languages) and a corpus created from Tamil novels and short stories from AUKBC, Anna university. The complete corpus consisted of 197,792 sentences. The Python extension to the morphological segmentation tool morfessor 2.0 was used for this experiment to perform the segmentation. The annotation data for Tamil language collated and released by Anoop Kunchukuttan in the Indic NLP Library was used as the semi-supervised input to the model (Smit et al. 2014; Virpioja et al. 2013). Other than this paper, there were no records of any other papers which used this segmentation approach for English - Tamil NMT.

3.1.2 Byte Pair Encoding

[Himanshu Choudhary et al 2018] proposed a neural machine translation technique using pre-trained word embeddings (BPEmb) to develop an efficient translation system that overcomes the OOV (Out Of Vocabulary) problem for languages which do not have much translations available online. This is the first work to apply BPE with word embedding on Indian language pair (English-Tamil) with NMT technique. Their model outperformed Google translator with a margin of 4.58 BLEU score. The base model which used Bidirectional LSTM with Adam (Vaswani et al., 2017) optimizer, Bahdanau (Bahdanau et al., 2014) attention mechanism and Fasttext word embeddings got 6.74 BLEU scores in English to Tamil direction. The models which used BPE along with the same configurations as the previous model got 8.14 and 8.33 BLEU scores when ran with 10k and 25k Byte pair merge operations respectively. So, BPE resulted in 1.59 BLEU score improvement. The submission of IIT, Patna for WAT 2018 multilingual translation task also included BPE in the model, but no results were reported regarding the performance increase in the model due to BPE.

3.1.3 Pre Trained Word Embeddings

Word embedding is a way of representing words on a vector space where the words having the same meaning have similar vector representations. They are a distributed representation for text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems.

[K Hans and Milton et al 2017] have used Word2Vec word embeddings along with the morphological segmentation. For the process of creating semantically meaningful word embeddings, a monolingual corpus of 569,772 Tamil sentences was used. The authors used the gensim word2vec toolkit to implement this word embedding process with a vector dimension of 100 and window size of 5. When using Word2vec, the 4-gram precision (BLEU-4)) is improved from 4.84 to 5.57. [Himanshu Choudhary et al 2018] have used FastText word vectors instead of WordToVec for their experiments. They used a vector size of 300 and got 0.56 (6.18 to 6.74) BLEU score improvement by using these pre-trained word embeddings.

3.2 Multilingual NMT

Since NMT performs poorly in low resource conditions and gives better results in high resource conditions, approaches like multilingual translation are proposed where more than 1 language pair is trained in the same model. When high resource language pairs and low resource language pairs are trained together,
parameters will be shared between them, and it helps low-resource pairs to learn better models compared to models trained separately.

The submission of IIT, Patna for multilingual Indic languages shared task at 5th Workshop on Asian Translation (WAT 2018) has implemented Indic languages to English and English to Indic languages multilingual models. This model included 7 Indic languages (Bengali, Hindi, Malayalam, Tamil, Telugu, Sinhalese and Urdu) and English. Multilingual NMT gave better performance in both directions for English-Tamil language pairs. In the Tamil to English direction, a BLEU score improvement of 10.84(11.58 to 22.42) is observed when using multilingual NMT model over NMT model trained only on English and Tamil language. 6.93(11.88 to 18.81) BLEU score improvement is observed in the opposite direction too.

3.3 Dataset Creation

NMT systems are very data hungry. So, using more and more data can increase its performance unless quality is very low. In the context of English - Tamil language pairs, the first reported efforts to create bilingual corpus is done by [Loganathan Ramasamy et al]. The parallel corpora which has 169871 sentence pairs, cover texts from bible, cinema and news domains. This is currently the largest dataset available publicly for English- Tamil NMT. Their research was focussing on using morphological processing to improve English - Tamil SMT. other than this, [Ramesh et al ] have tried to generate parallel sentences from comparable multilingual articles in Wikipedia. They have shown that the generated dataset improved BLEU scores on both NMT and phrase-based SMT systems for English - Tamil language pairs. The SMT model which is trained with sentences extracted from Wikipedia(along with EnTam dataset) got 0.55 (4.02 to 4.57) BLEU score improvement. NMT models showed an improvement of 0.50(4.53 to 5.03).

4. Discussion and Future Direction

To address the issues faced by translation involving low-resource and morphologically rich languages, many techniques have been proposed. Some of them have been applied and reported in respect to English - Tamil NMT systems and still there are many approaches which were proposed in research studies of other languages pairs, are not applied to English - Tamil NMT. When we consider sub word segmentation techniques, techniques like Byte Pair Encoding and morphological segmentation have been used with significant performance improvement. There are no reported studies which used other available techniques like WordPieces, SentencePieces, unigrams and character level models to English - Tamil NMT. With respect to word embeddings, WordtoVec and FastText are used but newer word embedding techniques like Glove and BERT are yet to be applied. Cross lingual word embeddings should also be tried out for English - Tamil NMT. When it comes to Multilingual NMT, there were different approaches/variants proposed in the literature which can also be used to improve the performance of English - Tamil NMT. Transfer learning techniques are used to improve English - Tamil NMT by using Hindi- English as the parent language pair, but the performance improvement got by it is not reported. Since Hindi is not related directly to Tamil, Experiments using similar languages to Tamil should be explored. Back translation is also another potential technique which can improve NMT performance and can be used as a technique to generate new datasets.

5. Conclusion

In this article, we have tried to present research studies done with regard to English - Tamil neural machine translation and the approaches they used to improve the quality of translation. Even though a sufficient amount of work is done, still there is room for improvement in translation quality. In this article, we first investigated the challenges in English-Tamil neural machine translation and then discussed the existing literature and the key techniques used by them to tackle those challenges. Finally, we have provided some
possible direction where further exploration is needed. We hope this article gave a small introduction to English - Tamil neural machine translation.

References


இல்லம் - அல்லாது தனிப்பிறரின் திரைகதா நிறுவனத்திற்கு

M. Julien Malard (Julien Malard)®, பார். பீமன் (Joel Harms)®, ஆட். பார்டன் (Jan Adamowski)®, சீமை - ஜிஞான (Hugo Melgar-Quinonez)®

® ஜன் மலர்ட் பிரான்சிஸ் கார், நாகர் பன் எண்ணெய் காம்புவன், காலா

® ஜோ ஹர்மஸ் பார்டன் பிரான்சிஸ் கார், நாகர் பன் எண்ணெய் காம்புவன், காலா

இல்லம் பிரிவுக் குழுக்களுக்கான இணையடிவாரியில் - julien.malard@mail.mcgill.ca; joel.harms@mail.mcgill.ca; jan.adamowski@mcgill.ca; hugo.melgar-quinonez@mcgill.ca

துறைமுகம்

இல்லாதை பிரிவேஸ்ட் திரைப்படங்கள் பிரிவுரைப்படாத அவர்களின் சூழ்நிலைக் காலம் முக்கியமானது. இம்முக பிரிவு பிள்ளைகளின் திரைகதா நிறுவனக் கூடியான விளக்கப்பட்டுள்ளது. இது பல்வேறு தமிழ்ப்பிறரின் ஆலவியல் பன்முக முறைகளின் முனையாக வளர்ந்தது. திரைநமரியுடன் இணைய்க்கப்பட்டுள்ள உலகப் பிரிவுரைப்படங்களின் ஆலவியல் பன்முக முறையை விளக்கப்பட்டுள்ளது. வலித்திரைநிலையில் இருந்தது தமிழ்ப்பிறரின் எழுத்துத் தாண்டை காலம்.
கற்டீரங்கு பராஷ்டிரரின் முன் சென்று இன்னொரு செரிய குறிப்பிட்டியர் (de Mooij என்றும் Jilani என்றும்). அவர்களின் தமது நல்லையை கூறுவதற்கு தனிப்பிட்டிய குறிப்பிட்டியர் ("துவுக கிளிக்குப்பு ஊடு")

நல்லையை பராஷ்டிரரின் முன் சென்று இன்னொரு செரிய குறிப்பிட்டியர் (de Mooij என்றும் Jilani என்றும்). அவர்களின் தமது நல்லையை கூறுவதற்கு தனிப்பிட்டிய குறிப்பிட்டியர் ("துவுக கிளிக்குப்பு ஊடு")

ஆகிய வகையில், குறிப்பிட்டியர் பராஷ்டிரரத்தின் செரிய குறிப்பிட்டியர் தன்மையடைவு வாராயியத்தாக குறிப்பிட்டிய குறிப்பிட்டியர் ("துவுக கிளிக்குப்பு ஊடு")

ஆகிய வகையில், குறிப்பிட்டியர் பராஷ்டிரரத்தின் செரிய குறிப்பிட்டியர் தன்மையடைவு வாராயியத்தாக குறிப்பிட்டிய குறிப்பிட்டியர் ("துவுக கிளிக்குப்பு ஊடு")

புர்வு குறிப்பிட்டியர் என்ன என்று கூறியுள்ளனர் மற்றும் உடனே சென்று இன்னொரு செரிய குறிப்பிட்டியர் ("துவுக கிளிக்குப்பு ஊடு")

மற்றும் உடனே சென்று இன்னொரு செரிய குறிப்பிட்டியர் ("துவுக கிளிக்குப்பு ஊடு")
உதாரண;.காக

வறி?

एக

மத

எண்: 8

இరடா

மயவ

ாBற

ர  உ:வாக4

/க

/ட(5

/ள

/ா(H

எ த ி (Wயி

லா( 5னர

Wயி

லா(நா$;

நட;.கான இனகா>W

ாBற

மச

/ட

ாதிரி

மானான;

இDத

மான;

மச

/ற

மூன்மாறு;

மச

லா

மச

45ட  இயB

/ட

மச

மச

மச

மச

/ள

/ள

/ள

/ள

/ள

/ள

/ள

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/ள.

/plaintask

/return 

/ நிலை  அடுத

/பிரிகளைக்  எ

/மாறு/ பிரிகளைக்  எ எனும் கிளப்ப செயல்விடத் திறன்னிலை  கிளப்ப இருளுக்கு நிலை நேரடையும்:

/ gibt அ  zurück

/இது குறுக்குச்சுட்டின்,  கிளப்ப செயல்விடத் திறன்னிலை  செயல்விடத் திறன்னிலை. நிலை  குறுக்குச்சுட்டின், அதை குறுக்குச்சுட்டின்

/if  ஒலிய > 0:

/ pass

/ேட்டிலை  அடுத

/ேட்டிலை >  விளக்கம்:

/ விள
டிவிக்க நூற்றாண்டுகளாக உருண்டையட்ட சக்தியை வழங்கப்படும் பாணிகளின் பதிவுகைப்படுத்தல் (லார்ல/EARL, EARLY/earley) பயணியப்படுத்தலாக நாட்டுமுகராச்சிய ஆராய்ச்சிகளின் கலனிகள் குறிப்பிட்டுள்ளன. உதவும் 2-டக்க வாயில் வழங்குவதற்கு ஏற்படும் விளக்கம் மற்றும் மற்றும் பொருளாக்கம் பெற்றுள்ளது. ஜெலியோ போன்ற பெரும் தொடர்புபட்ட பெயர் விளக்கம் நேரத்தான தொடர்புபட்ட பெயர் விளக்கம்.

<table>
<thead>
<tr>
<th>நம்பியது விளக்கம் முக்கியமான</th>
<th>அந்ததொடர் விளக்கம் முக்கியமான</th>
<th>நம்பியது விளக்கம் முக்கியமான</th>
<th>அந்ததொடர் விளக்கம் முக்கியமான</th>
</tr>
</thead>
<tbody>
<tr>
<td>if_stmt: cond &quot;நத்தால்&quot; : suite</td>
<td>if_stmt: &quot;if&quot; cond &quot;.&quot; suite</td>
<td>if_stmt: &quot;if&quot; cond &quot;.&quot; suite</td>
<td>if_stmt: &quot;if&quot; cond &quot;.&quot; suite</td>
</tr>
<tr>
<td>cond: var &quot;==&quot; expr</td>
<td>cond: var &quot;==&quot; expr</td>
<td>cond: var &quot;==&quot; expr</td>
<td>cond: var &quot;==&quot; expr</td>
</tr>
<tr>
<td>expr: num</td>
<td>var</td>
<td>expr: num</td>
<td>var</td>
</tr>
<tr>
<td>pass_stmt: &quot;pass&quot;</td>
<td>pass_stmt: &quot;pass&quot;</td>
<td>pass_stmt: &quot;pass&quot;</td>
<td>pass_stmt: &quot;pass&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>குறிப்பிட்டு விளக்கம் முக்கியமான</th>
<th>குறிப்பிட்டு விளக்கம் முக்கியமான</th>
<th>குறிப்பிட்டு விளக்கம் முக்கியமான</th>
<th>குறிப்பிட்டு விளக்கம் முக்கியமான</th>
</tr>
</thead>
<tbody>
<tr>
<td>cond: var: குறிப்பிட்டு விளக்கம் முக்கியமான</td>
<td>cond: var: குறிப்பிட்டு விளக்கம் முக்கியமான</td>
<td>cond: var: குறிப்பிட்டு விளக்கம் முக்கியமான</td>
<td>cond: var: குறிப்பிட்டு விளக்கம் முக்கியமான</td>
</tr>
<tr>
<td>var: pi</td>
<td>num: குறிப்பிட்டு விளக்கம் முக்கியமான</td>
<td>num: குறிப்பிட்டு விளக்கம் முக்கியமான</td>
<td>num: குறிப்பிட்டு விளக்கம் முக்கியமான</td>
</tr>
<tr>
<td>suite pass_stmt</td>
<td>suite pass_stmt</td>
<td>suite pass_stmt</td>
<td>suite pass_stmt</td>
</tr>
</tbody>
</table>

2-டக்கம் க: எளியாக விளக்கப்பட்ட முயற்சிகள் நோய் குறிப்பிட்டு விளக்கம் முக்கியமான.
<table>
<thead>
<tr>
<th>வர்ண விளக்கம்(வகு): பாய் = 3.4815992653</th>
</tr>
</thead>
<tbody>
<tr>
<td>வர்ப் விளக்கம்(வகு): பாய் = 3.4815992653</td>
</tr>
<tr>
<td>நூற்றாண்டு (சேதே, நடுவில்): கல்கா, சுமார் = 1848</td>
</tr>
<tr>
<td>குடியாரப் (சைந்தனை): கலை, புதியார் = சூடியார்</td>
</tr>
<tr>
<td>பிரிவுக்கு(சைந்தனை): கலன், புதியார் * கலன், சுமார் ** புதியார்</td>
</tr>
<tr>
<td>அரைக்காற்றும் (சைந்தனை): கலன், புதியார்</td>
</tr>
<tr>
<td>தொடர்வற்றும் (சைந்தனை): கலன், புதியார்</td>
</tr>
</tbody>
</table>

while_stmt : "while" test ":." suite [ "else" ":." suite ]

while_stmt : test ":." suite [ "else" ":." suite ]

2. முதலாம் பகுதி

முதலாம் பகுதியில் பல்வேறு முழுக்கு படிகங்களைக் குறிக்கும் ஊடைகள் உள்ளன. இவைகள் முழுக்கு படிகங்களின் தொகுப்பான குறிகள் குறிக்கின்றன. இவைகள் முழுக்கு படிகங்களின் தொகுப்பானது நேர்மாற்றப்பட்ட நேர்மாற்றத்தைக் குறிக்கிறது. இவைகள் குறிக்கின்றது முழுக்கு படிகங்களின் தொகுப்பானது நேர்மாற்றப்பட்ட நேர்மாற்றத்தைக் குறிக்கிறது. இவைகள் குறிக்கின்றது முழுக்கு படிகங்களின் தொகுப்பானது நேர்மாற்றப்பட்ட நேர்மாற்றத்தைக் குறிக்கிறது.


Speech Embedding with Segregation of Para-linguistic Information for Tamil Language: a survey

Anosha Ignatius  
*dept. Computer Science and Engineering*  
*University of Moratuwa*  
Sri Lanka  
anoshai@uom.lk

Uthayasanker Thayasivam  
*dept. Computer Science and Engineering*  
*University of Moratuwa*  
Sri Lanka  
rtuthaya@cse.mrt.ac.lk

**Abstract**—Deep Neural Networks (DNN) based speech embedding techniques have shown significant performance in speech processing applications. However, the presence of para-linguistic information such as speaker characteristics, accent, pronunciation, and emotion expression causes performance degradation in speech recognition where only the linguistic content is needed. Over the years many techniques have been proposed to address this problem by disentangling the para-linguistic content from the speech signal to learn better representations. The most common approach is to provide speaker level information at the input of acoustic model. In the case of low resource conditions when only a limited amount of transcribed speech data is available, unsupervised speech representation learning approach is adopted. A detailed study of research work related to disentangled speech representations is presented in this paper. Speech embedding techniques for Tamil Language are also discussed.

**Index Terms**—speech processing, linguistic, para-linguistic information, speaker characteristics

I. INTRODUCTION

Speech is a time-varying signal carrying multiple layers of information: linguistic information and para-linguistic information. Linguistic information refers to the meaning of the words being spoken while para-linguistic information refers to the residual information remaining after removing the verbal content from speech. Para-linguistic information covers the manner of speaking varied with the accent, pronunciation, emotional state and speaker traits. Speech processing applications that are focused on the linguistic content such as Automatic Speech Recognition (ASR) and Speech Intent Recognition have yielded significant performance over the time. However, their performance can be greatly compromised due to mismatch between testing and training conditions. It is because they are subjected to variations in the manner of speaking caused by speaker characteristics, emotional expressions and environmental differences. Therefore, carrying out an extensive research on this area is of great importance for developing techniques that compensate for the speaker variability and improve the robustness of the systems. Several studies have investigated DNN based speech embedding models for removing the para-linguistic information from the speech signal while retaining the linguistic content.

DNN based speech recognition model is typically trained using acoustic feature vectors representing both the time domain and frequency domain information in the speech signal. Acoustic feature vectors are extracted using various feature extraction techniques such as Mel Frequency Cepstral Coefficients (MFCC), Linear Predictive Codes (LPC) and Perceptual Linear Prediction (PLP) [1]. MFCC features are most commonly used as input features in speech processing applications. Existing solutions for normalizing the variations caused by para-linguistic information incorporate speaker level information by augmenting the acoustic features with speaker representing vectors. This helps in disentangling the underlying speaker information in the speech signal by learning better representations. Vectors representing the speaker-specific information are obtained using a separate model. It involves mapping the variable length speech utterance to a fixed dimensional vector that captures speaker features. Prior studies have explored mainly two types of such techniques: i-vector approach [2] which is the conventional one and DNN based speaker embedding techniques. By providing speaker representations to the DNN as additional input features, DNNs are trained to be aware of the presence of speaker information. This approach is referred to as speaker aware training. Feature augmentation enables the acoustic model to normalise speaker effects, thus leading to a better generalization of the model to unseen conditions.

Under low resource conditions where a significant amount of labeled training data is not available, it is difficult to separate linguistically irrelevant speaker information encoded in the speech features. Supervised acoustic modelling relies on large amount of transcribed speech data to ensure the robustness against speaker variability. Thus, in the case of low resource scenario, unsupervised speech representations that separate speaker traits from linguistic content are desired to address
this problem. Such representations learnt with an available large data set of unlabeled speech recordings can be used to learn an acoustic model with only a small amount of labeled training data. Tamil language comes under the low resource category and only a few studies have explored the problem of speaker variations.

The rest of the paper is organized as follows: Section II gives a detailed account of speaker embedding techniques, and discusses various methods for augmenting acoustic features with speaker vectors. Section III explores speech representations obtained using unsupervised learning approach and Section IV discusses Tamil speech recognition. The paper is concluded in Section V, with discussions in this research area.

II. SPEAKER AWARE TRAINING

ASR DNNs can be made invariant to speaker variability by providing speaker information to the DNN using speaker representing vectors. Augmenting the acoustic features with speaker vectors transforms the input features to a speaker normalised space.

A. Speaker Embedding

Speaker embedding encodes long term speaker characteristics that are difficult to learn with acoustic models using short term features. i-vector is an effective method to map a variable length speech segment to a low dimensional representation that captures speaker features. i-vector extraction involves Universal Background Model (UBM) and Total variability matrix. UBM, a speaker independent Gaussian Mixture Model (GMM) identifies and clusters similar acoustic features together. The total variability matrix represents the basis of the total variability space which models both the speaker variability and channel variability subspaces in a common low dimensional space. UBMs and the total variability matrix are learnt in an unsupervised manner using expectation maximization algorithm. High-dimensional statistics from the UBM are then mapped into a low-dimensional i-vector. i-vectors have been extensively used in speaker identification and verification tasks. i-vectors are widely used for speaker normalization in ASR systems as well.

Speaker representation vectors can also be extracted using DNN based embedding techniques. In DNN based speaker embedding, a DNN which is trained to discriminate between speakers is used to extract the speaker representing vectors at the bottleneck layer. Recent DNN embeddings such as x-vectors extracted from a Time Delay Neural Network (TDNN) with a pooling layer [3] and h-vectors extracted using a hierarchical attention network [4] achieved better performance in speaker recognition tasks.

B. Speaker Normalisation

Speaker normalisation using feature augmentation techniques involves incorporating the speaker level information extracted as fixed dimensional vector in the input. It helps the DNN learn to normalize the speaker effects, thereby improving the performance. Several forms of feature augmentation include directly appending the speaker vectors to acoustic features and providing transformed features to the DNN during training.

The conventional i-vector based augmentation method uses i-vectors as additional input features. For each frame of a given utterance, same utterance level i-vector is concatenated with the acoustic features. Providing speaker information at the input enables the DNN to normalize the signal and make it invariant to speaker effects [5], [6]. Sri Garimella et al. proposed passing the i-vectors through a nonlinear hidden layer before combining them with the acoustic features [7]. In this model, connectivity from to the rest of network i-vectors is restricted to improve the robustness of the model. Using a nonlinear hidden layer to transform the i-vectors showed improvement over directly appending the to the acoustic features.

Two types of feature mapping neural networks: ivecNN and adaptNN are presented in the paper by Miao et al. where i-vectors are used as additional inputs to project acoustic features into a speaker normalized space [8]. In ivecNN, a bias vector is estimated and added to the original features making the resulting feature space speaker independent. The network takes i-vectors as the output and generates a feature shift as the output. The output is added to the acoustic features and fed to the DNN acoustic model. In adaptNN, multiple adaptation layers are used under the initial DNN acoustic model where each adaptation layer except the last one appends the i-vector to its output. By incorporating the i-vectors, the adaptation layers convert the original DNN input into more speaker independent features. Experimental results showed that these two networks achieve better performance over the original DNN with acoustic features.

Xiangang Li et al. investigated the idea of augmenting acoustic features with d-vectors, speaker embedding learnt using long short-term memory (LSTM) recurrent neural networks (RNNs) [9]. They proposed cross-LSTMP to conduct speaker recognition and speech recognition simultaneously. cross-LSTMP consists of two LSTMs, senone-LSTM for classifying senones and speaker-LSTM for classifying speakers. In each frame, previous activations of speaker-LSTM are fed into the senone-LSTM while the previous activations of senone-LSTM are into the speaker-LSTM to make the network speaker aware. Conducted experiments indicated that the proposed approach can effectively improve the performance by compensating for the speaker invariability in ASR.
Xiaodong Cui et al. proposed an embedding based speaker adaptive training approach [10] where speaker vectors are mapped to layer dependent element-wise affine transformations through a control network. Resulting affine transformations are applied to the internal feature representations at the outputs of the selected hidden layers in the acoustic model to facilitate speaker normalization. This approach outperformed feature augmentation with i-vectors.

Speaker Aware Speech Transformer, a standard speech transformer [11] with a speaker attention module (SAM) is presented in the paper by Zhiyun Fan et al. [12]. SAM includes a speaker knowledge block that consists of a group of i-vectors extracted from the training data and for each frame SAM generates a soft speaker embedding. As there are similarities among different speakers, speaker vector can be represented as a linear combination of a set of basic speaker representations. Thus, given a speech utterance, similarity of the acoustic feature vector and each i-vector from the group of basic speakers in speaker knowledge block is computed with the attention mechanism to obtain the weight for each basic i-vector. The soft speaker embedding is extracted as the weighted sum of the basic i-vectors and a weighted combined speaker embedding vector is fed to decoder. This helps the model to normalize the speaker variations and leads to better generalization to unseen test speakers. Similar approach where attention mechanism is used to select the relevant speaker i-vectors for each frame from the memory is proposed by Jia Pan et al. [13] Speaker aware speech transformer proved to be more effective than other feature augmentation techniques using i-vectors.

III. DISENTANGLLED SPEECH REPRESENTATIONS WITH UNSUPERVISED LEARNING

Unsupervised learning method uses large amount unlabeled data to learn useful speech representations that can be incorporated into several downstream applications under low resource conditions. Auto encoders is one such network that involves the reconstruction of its inputs. Auto encoders consists of an encoding network that extracts a latent representation and a decoding network that tries to reconstruct the original data using the latent representation. By applying constraints, the network is made to learn a latent representation that discards irrelevant para-linguistic information while preserving the information necessary for perfect reconstruction.

Factorized Hierarchical Variational Auto encoder (FHVAE) [14] is proposed by Wei-Ning Hsu et al. to learn disentangled representations from sequential data. It can be used to separate linguistic content and speaker information in speech signal in an unsupervised way. The FHVAE learns to factorize sequence-level and segment-level attributes of speech into different latent variables [15], [16]. Speaker identity affects the speech features at sequence-level while phonetic content affects the speech features at utterance level. Hence, sequence-level attributes show a small amount of variation within an utterance and segment-level attributes show similar amounts of variation within and across utterances. Based on this, different sets of latent variables are generated. Based on this, different sets of latent variables are generated. Latent variable that encodes the linguistic content while discarding the speaker characteristics is used in robust speech recognition task.

Jan Chorowski et al. presented a Vector Quantized Variational Auto encoder (VQ-VAE) that learns a representation to capture high level semantic content while being invariant to the underlying speaker information in the signal [17]. It conditions the decoder on speaker identity which frees up the encoder from having to capture speaker features, thus resulting in a more speaker invariant representation. Best representations are obtained when MFCC features are used as inputs and raw waveforms are used as targets. Waveforms are reconstructed using a WaveNet decoder that combines auto-regressive information about past wave samples, speaker information and latent information extracted by the encoder. VQ-VAE yielded a significant performance in the phonetic unit discovery task indicating a better separation between phonetic content and speaker information.

Another unsupervised objective, Auto-regressive Predictive Coding (APC) [18]–[20] is proposed by Yu-An Chung et al. can be used as a pre-training approach for learning meaningful and transferable speech representations. It is trained to predict the spectrum of future frames n steps ahead of the current one using the past values to infer more global structures in speech rather than exploiting local smoothness of the speech signal. Unlike the previously discussed research studies which attempted to discard the irrelevant information, APC model aims to preserve as much information such that the extracted representations could be used for a variety of downstream tasks. Experiments conducted using intermediate representations obtained from APC model on different speech applications such as speaker verification, phone classification and automatic speech recognition indicate that different levels of speech information are captured by the APC model at different layers. Lower layers contain more speaker information while the upper layers provide more phonetic content. Combination of the internal representations extracted across different layers could be beneficial in learning disentangled representations for speech recognition tasks.

The aforementioned research studies show that unsupervised learning method can be effectively applied on large amounts of unlabeled data to extract speaker invariant features that helps in improving the robustness of low resource speech recognition systems.
IV. TAMIL SPEECH RECOGNITION

Improving speech recognition in Tamil language is an active research field and it is a challenging task due to the unavailability of a large corpus of Tamil speech. Though, many research studies have attempted to build a reasonably good speech recognition systems [21], [22]. Few studies have investigated the variations caused by speaker characteristics. Akila et al. addressed the effects of speech rate variability by applying time normalization to the speech signal [23]. The speech signal is categorized as slow, normal and fast speech using features such as the sound intensity level and time duration. If the speech rate is either slow or fast, time normalisation is applied. In the paper by Madhavaraj et al. speaker adaptive training is used with Maximum likelihood linear transformation to improve the recognition [23].

V. SUMMARY

The study presented an overview of various techniques that suppress the effects of para-linguistic information in speech recognition systems. Disentangling the linguistically irrelevant information is important to alleviate the problem of mismatch between training and testing conditions. Existing solutions were studied under two categories: speaker embedding based feature augmentation and unsupervised speech representation learning. Incorporating the speaker information into the acoustic model through speaker embedding have resulted in improved recognition rate but such supervised acoustic models require large amount of labeled training data for a more robust system. In case of low resource scenario, unsupervised learning method can be adopted where speech representations learnt from a large-scale unlabeled data are used for downstream tasks with a limited amount of labeled training data. This is proved to be a powerful approach to achieve better generalisation for acoustic models. Speech Recognition systems for Tamil language which lacks speech data could exploit these methods to achieve better performance.

ACKNOWLEDGMENT

This research was supported by Accelerating Higher Education Expansion and Development (AHEAD) Operation of the Ministry of Education funded by the World Bank.

REFERENCES


