National Languages Processing Centre
University of Moratuwa

PERIYAR UNIVERSITY
SALEM
Wisdom Maketh World

உத்தமம்
INFITT
உலகத் தமிழ்த் தகவல் தொழில்நுட்ப மன்றம்
International Forum for Information Technology in Tamil

உலகத்தமிழ் தகவல் தொழிற்நுட்ப மன்றம் (உத்தமம்)

19வது தமிழ் இணைய மாநாடு (மெய்நிகர்) -2020

19th Tamil Internet Conference (Virtual) -2020

மாநாட்டு நிகழ்வின் இணை ஒருங்கிணைப்பாளர்கள்

இலங்கை மொரட்டுவ பல்கலைக்கழகத்தின் தேசிய மொழிகள் ஆய்வு மையம்

மற்றும்

பெரியார் பல்கலைக்கழகம் சேலம், இந்தியா

டிசம்பர் : 11 - 13, 2020

மாநாட்டுக் கட்டுரைகள்

Conference Proceedings

தொகுப்பு

முனைவர் வாசு அரங்கநாதன்
பென்சில்வேனியா பல்கலைக்கழகம்.

பதிப்பு

உலகத் தமிழ்த் தகவல் தொழில்நுட்ப மன்றம், உத்தமம்.

9 772313 488707

மெய்நிகர் மாநாடு

மாநாட்டுக் கட்டுரைகள்

தொகுப்பு
பேராசிரியர் வாசு அரங்கநாதன்
தலைவர், மாநாட்டு நிகழ்ச்சிகள் குழு
பென்சில்வேனியப் பல்கலைக்கழகம்
பிலடெல்பியா, அமெரிக்கா

பதிப்பு
உலகத் தமிழ்த் தகவல் தொழில் நுட்ப மன்றம் (உத்தமம்)
அமெரிக்காவின் கலிபோர்னியா மாநிலத்தில் இலாப
நோக்கற்ற நிறுவனமாகப் பதிவுசெய்யப்பட்ட நிறுவனம்

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

ISSN: 772313 - 488707

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

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

Table of Contents

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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

December 11th, Friday between 5:00PM and 8:30PM IST
Inauguration: MC: **Ms. Sandana Vellien, Educator,** Seewa Bappoo State Secondary School, Mauritius Welcome Speech: Dr. Arul Veerappan, Chair, INFITT Introduction, CPC: Dr. Vasu Renganathan, Chair, CPC, INFITT Professor Gihan Dias, Co-Chair, University of Moratuwa Inaugural Speech: Professor Ponnavaikko, Founder, INFITT Keynote: Professor M. A. Anandakrishnan, Founder, INFITT Greetings messages: Professor P. Kolandaivel, Vice-Chancellor, Periyar University, Salem Professor C. Subramaniam, Former Vice-Chancellor, The Tamil University, Thanjavur. Release of Conference Proceedings – Conference Program Committee and members of the Inaugural meeting. Vote of Thanks: Professor K. Kalyanasundaram, Former Chair, INFITT
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 12nd, 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

Session II (5:30 to 6:00 PM)
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.
Session III (6:00 to 6:30PM)
Chair: **Dr. Arul Veerappan, Chair, INFITT**. **Speaker: அகிலன்,** செம்மொழி நிறுவனம், சென்னை Paper: தரவக வழித் தமிழ்ச் செவ்வியல் நூல்கள் ஆய்வு: சிக்கல்களும் தீர்வுகளும்
Session IV (6:30 to 7:00PM)
Chair: Dr. M. Ganesan, Annamalai University **Speaker: Dr. K. Rajan,** Annamalai University Paper: Application of Letter Successor Varieties in Tamil Morphological Analysis.
Session V (7:00 to 7:30PM)
Chair: Prof. K. Kalyanasundaram, Switzerland **Speaker: Muthiah Annamalai,** Ezhilang Paper: Algorithm to Correct Missing Pulli –Signs in Printed Tamil Text:
Session VI (7:30 to 8:00PM)
Chair: Maniam, S., Singapore. **Speaker: ஜூலியன் மலார்ட்,** மக்கில் பல்கலைக்கழகம் Paper: லஸ்ஸி - உங்கள் தாய்மொழியில் நிரலாக்கம். ஜூலியன் மலார்ட், ஹா. யோல், ஆ.யான், மெ-கீ. ஊகோ உயிர் வள பொறியியல் துறை - மக்கில் பல்கலைக்கழகம், கனடா, உலக உணவு பாதுகாப்பு நிலையம், மக்கில் பல்கலைக்கழகம், கனடா
Session VII (8:00 to 8:30PM)
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

December 13th, Sunday between 5:00PM and 8:30PM IST
Keynote Session I (5PM to 5:30PM)
Chair: Professor Deivasundaram, Chennai **Speaker: Rajendran Sankaravelayuthan**, Amrita Vishwa Vidyapeetham, Coimbatore. Paper: Computational Analysis of Nominal Compounds in Tamil.
Session II (5:30 to 5:55PM)
Chair: **K. Sarveswaran, Sri Lanka** **Speaker**: **T. Shrinivasan** **Topic**: Tamilnayavaani – Integrating TVA Open-Source Spellchecker with Python. T. Shrinivasan, Nityha Duraisamy, Ashok Ramachandran, Manickkavasakam, Arunmozhi, and A. Muthiah.
Session III (5:55 to 6:20 PM)
Chair: Elantamil, Malaysia **Speaker: ராமன் விமலன்**, ஆசிய மொழிகள் மற்றும் பண்பாடுகள் துறை, தேசியக் கல்விக் கழகம் நனயாங் தொழில்நுட்பப் பல்கலைக்கழகம், சிங்கப்பூர். Paper: கலவை முறைக் கற்றலில் தொழில்நுட்ப வளங்களின் பன்முகப் பயன்பாடு
Session IV (6:20 to 6:45PM)
Chair: Rama Suganthan, Tamilnadu, India **Speaker: பரதன் தியாகலிங்கம்** **Paper:** மாத்திரை பார்வையில் குறள், பரதன் தியாகலிங்கம், முத்து அண்ணாமலை.
Session V (6:45 to 7:10PM)
Chair: Shrinivasan, T., Chennai **Speaker: Malaikkannan Sankarasubbu, Saama Technologies AI Research Lab** **Paper: SymSpell and LSTM based Spell Checkers for Tamil**
Session VI (7:10 to 7:35PM)
Chair: Sakthivel Ramasamy, Chennai **Speaker: Sathiya Mahadevan** **Paper**: Generation and Parsing of Number to Words in Tamil: Mutiah Annamalai and Sathiya Mahadevan

Session VII (7:35 to 8:00PM)
Chair: Muthu Annamalai, California **Paper**: A Survey on Neural Machine Translation for English-Tamil Language pair **Speaker: B.Janarthanasarma, T.Uthayasanker**, Department of Computer Science and Engineering, University of Moratuwa {janarthanasarma.13,rtuthaya}@cse.mrt.ac.lk
Valedictory Function (8:00 to 8:30PM)
Welcome speech: Mr. Maniam, Singapore (Executive Director, INFITT) INFITT Activities: Dr. Arul Veerappan (Chair, INFITT) INFITT going forward: Professor M. A. Anandakrishnan, Founder, INFITT Valedictory speech: Prof. Ponnavaikko, Founder, INFITT Next INFITT Co-Sponsor: Dr. G. Viswanathan, Chancellor, Vellore Institute of Technology, Vellore. Concluding thoughts on Tamil NLP research: Professor Gihan Dias, Sri Lanka Vote of Thanks: M. A. Ganesan, Vice Chair, INFITT

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

முனைவர் அருள் வீரப்பன்

தலைவர், உத்தமம்

அன்புகெழுமிய உத்தமம் நிறுவன உறுப்பினர்களுக்கு வணக்கம். உத்தமம் நிறுவனத்தின் தலைமை பொறுப்புக்கு என்னைப் பணித்தமைக்கு எனது நன்றிகள். இந்த இக்கட்டான சூழலிலும் நமது நிறுவனத்தின் பத்தொன்பாவது வருடாந்திர மாநாடு நடைபெறுகிறது என்பதை எண்ணி மகிழ்ச்சியடைகிறேன். இம்மாநாட்டை நடத்தத் தொடக்க காலத்திலிருந்து என்னோடு பணிபுரிந்து வந்த செயற்குழு உறுப்பினர்கள் அனைவருக்கும் எனது நன்றிகள் உரித்தாகுக. குறிப்பாக நமது செயல் இயக்குனர் திரு. மணியம் மற்றும் துணைத் தலைவர் பேராசிரியர் மா. கணேசன் ஆகியோர் தொடர்ந்து நமது நிறுவனத்தின் அடுத்தக் கட்டப் பணிகளைச் செவ்வனே நடத்தி வந்தமை போற்றற்குரியது. உத்தமம் நிறுவனத்தின் பதினெட்டு ஆண்டுகள் வரலாற்றில் முதல் முறையாக மெய்நிகர் வழி இம்மாநாட்டை நடத்துவது வியக்கத்தக்கது. என்றும் போல் இம்மாநாட்டிலும் பல ஆய்வாளர்கள் தங்களுடைய ஆய்வுகளைப் படைக்கவிருக்கின்றனர். அவர்களின் ஊக்கம் நம்முடைய செயற்பாடுகளுக்குப் புதுப்பொலிவு கொடுக்கிறது என்றே கூறவேண்டும். தமிழ்க்கணினி ஆய்வாளர்கள் தங்களின் படைப்புகளை உத்தமம் நிறுவனம் என்னும் குடையின் கீழ் படைக்க விரும்புகிறார்கள் என்று எண்ணும் போது உத்தமம் நிறுவனத்தின் பெருமை கூடுகிறது என்றே கூறவேண்டும். இம்மாநாட்டை நம்மோடு நடத்த முன்வந்த இலங்கையின் மொராத்துவா பல்கலைக்கழகத்துக்கும் பெரியார் பல்கலைக்கழத்துக்கும் நாம் நன்றி கூறக் கடமைப்பட்டிருக்கிறோம். இலங்கையிலிருந்தும் பல படைப்புகள் வந்துள்ளன. அத்தோடு பேராசிரியர் ஜிகான் தலைமையில் இலங்கையில் தமிழ்க்கணினி ஆய்வு பற்றிய ஒரு தனி நிகழ்வும் நடக்கவுள்ளது. இம்மாநாட்டு நிகழ்ச்சிகள் குழுவுக்குத் தலைமையேற்று நடத்தும் பேராசிரியர் வாசு அரங்கநாதன் மற்றும் ஜிகான் டையாஸ் ஆகியோருக்கு நன்றிகள். இம்மாநாட்டில் இன்னொரு சிறப்பாக உத்தமம் நிறுவனத்தின் மேனாள் தலைவர்கள் பேராசிரியர் கு. கல்யாணசுந்தரம், அப்பாசாமி முருகையன், வாசு அரங்கநாதன் மற்றும் இளந்தமிழ் ஆகியோரும் தொடர்ந்து நம் நிறுவனத்துக்குத் தங்களின் ஒத்துழைப்பையும் ஊக்கத்தையும் நல்கிவருகிறார்கள். உத்தமம் நிறுவனத்தின் வளர்ச்சியை அடுத்த ஆண்டும் தொடர்ந்து எடுத்துச் செல்லப் பேராசிரியர் மா. கணேசன் அவர்களின் தலைமையை நாம் ஆவலோடு எதிர்பார்ப்போம். பேராசிரியர் பொன்னவைக்கோ மற்றும் பேராசிரியர் அனந்தகிருஷ்ணன் ஆகியோரின் தொடர்ந்த வழிகாட்டுதல் நம் நிறுவனத்துக்கு என்றென்றும் கிடைப்பது மகிழ்வைத் தருகிறது. மாநாடு சிறக்க வாழ்த்துகள். உத்தமம் மலையென உயரட்டும்.

அன்புடன்,

அருள் வீரப்பன், நியூயார்க், அமெரிக்கா, டிசம்பர் 09, 2020

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

பேராசிரியர் மு.ஆனந்தகிருஷ்ணன்
பத்மஸ்ரீ
மேனாள் துணைவேந்தர், அண்ணா பல்கலைக்கழகம்
மேனாள் தலைவர், ஐஐடி கான்பூர்
மேனாள் தலைவர், உத்தமம்

வாழ்த்து மடல்

தமிழ் இணைய மாநாடு 1999 ஆம் ஆண்டு சென்னையில் வெற்றிகரமாகவும் பயனுள்ள வகையிலும் நடைபெற்ற பின் தமிழ் இணைய மாநாடுகள் உலகில் தமிழர் அதிகமாக வாழும் நாடுகளில் நடத்தப்பட்டு , இணையத்தில் தமிழ் மொழியின் பயன்பாடு அதிகரித்து தற்போது தமிழ் இணைய மாநாடு ZOOM காணொளி மூலமாக நடைபெருகின்றது. இம்மாநாட்டை இலங்கை மோருட்டுவா பல்கலைக் கழகமும், இந்தியாவிலிருந்து சேலம் பெரியார் பல்கலைக் கழகமும் சேர்ந்து டிசம்பர் 11 முதல் 13 வரை நடத்துகின்றன.

சிரமங்களை பொருட்படுத்தாமல் இவ்வாறு நடத்த முடிவெடுத்தது பாராட்டுக்குறியது. எப்போதும்போல சிறந்த கட்டுரைகள் விவாதிக்கப்படும் என நம்புகிறேன். பல்வேறு பொறுப்புகளுக்கிடையே இம்மாநாட்டை செவ்வனே நடத்த பொறுப்பேற்றுள்ள பேராசிரியர் வாசு ரங்கநாதன் அவர்களுக்கு நன்றியும் பாராட்டுகளையும் கூற கடமை பட்டுள்ளோம்.

மாநாட்டின் அமைப்பு, பொருள், ஆய்வுக்கட்டுரைகளின் தரம், ஒவ்வொரு ஆண்டும் உயர்ந்து கொண்டே வந்துள்ளன. வேகமாக வளரும் கணித்தமிழ் பயன்பாட்டுக்கேற்றவாறு தீர்வு காணவேண்டிய சிக்கல்களை விவாதிக்க இது ஒரு நல்ல வாய்ப்பு. என்னுடைய அனுபவத்தில் ஒவ்வொரு மாநாட்டிலும் கட்டுரைகளுக்கும், சொற்பொழ்வுகளுக்கும் அப்பால் ஆர்வலர்கள் நெருங்கி பழகவும், இளையதலைமுறைகளை சந்திக்கும் வாய்ப்புகளை உருவாக்குதலும் மிக சிறந்த பயனாக கருதுகிறேன்.

மு.ஆனந்தகிருஷ்ணன்

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

TIC 2020 பத்தொன்பதாவது தமிழ் இணைய மாநாட்டிற்கு வாழ்த்து

மு.பொன்னவைக்கோ

முதல் தமிழ் இணைய மாநாடு, 1997-ஆம் ஆண்டு, மே மாதம் சிங்கப்பூரில் 'தமிழ் இணையம் 97' என்ற பெயரில் நடைபெற்றது. Internet என்னும் ஆங்கில சொல்லுக்கு 'இணையம்' என்னும் தமிழ்ச்சொல்லை கணித்தமிழ் உலகிற்கு வழங்கிய பெருமை சிங்கப்பூரைச்சாரும். இரண்டாவது தமிழ்இணைய மாநாடு, 'தமிழ் இணையம் 99' என்ற பெயரில் சென்னையில் 1999-ஆம் ஆண்டு பிப்ரவர் மாதம் நடைபெற்றது. இம்மாநாட்டை அடுத்து 'தமிழ் இணையம் 2000' மாநாடு இலங்கையில் நடத்தத் திட்டமிடப்பட்டு இருந்தது. அந்த மாநாட்டின் முன்னேற்பாட்டுக் கூட்டம் 2000-ஆம் ஆண்டு நவம்பர் மாதம் இலங்கையில் நடைபெற்றது. அந்தக் கூட்டத்தில் சுவிட்சர்லாந்து நாட்டிலிலுள்ள முனைவர் கல்யாணசுந்தரம் அவர்கள், உலகஅளவில் ஒரு இணையத்தமிழ் ஆய்வுக்குழு அமைக்க வேண்டுமென வரைந்து அனுப்பியிருந்த திட்டத்தை திரு.அருண்மகிழ்நன் முன்மொழிந்தார். அந்தக் கலந்துரையாடலில் பிறந்ததுதான் 'உத்தமம்' என்னும் உலகத்தமிழ்த் தகவல் தொழில் நுட்பமன்றம். இம்மன்றத்திற்குப் பெயரிடும் பெருமை எனக்குக் கிட்டியது நான் பெற்ற பேறு. அந்த மாநாட்டை அடுத்து சிங்கப்பூரில் 2000-ஆம் ஆண்டு ஜூலை மாதம் 22-24-ஆம் நாட்களில் நடைபெற்ற தமிழ் இணைய மாநாட்டில் 'உத்தமம்' (INFITT) தொடங்கி வைக்கப்பட்டது. இணையத்தமிழின் ஆய்விற்காக உத்தமத்தில் தமிழ்க் கலைச்சொல் ஆக்கல், யூனிகோடு தமிழ் (UNICODE Tamil) வளர்ச்சி, இணையதள தமிழ் முகவரி வடிவமைத்தல், தமிழ் வரிவடிவக் குறியீட்டுத் தரப்பாடு, ஆங்கில வரிவடிவத் தமிழ்த் தரப்பாடு, தமிழ் எழுத்துரு அறிதல் (Tamil OCR), லினக்ஸில் தமிழ் (Tamil in Linux) ஆகிய பணிகளுக்காக ஏழு ஆய்வுப் பணிக்குழுக்கள் (Working Groups) நிறுவப்பட்டன. பணிக்குழுக்களின் செயற்பாடுகள் பற்றியும் கணித்தமி.ழ் மற்றும் இணையத்தமிழ் வளர்ச்சி பற்றியும் ஒவ்வொரு ஆண்டும்

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

உத்தமம் நடத்திய தமிழ் இணைய மாநாடுகளில் கலந்தாய்வு செய்யப்பட்டுள்ளன. இதுவரை (1997-லிருந்து 2014-வரை) சிங்கப்பூர், தமிழ்நாடு,, மலேசியா, அமெரிக்கா, ஜெர்மனி, புதுச்சேரி ஆகிய நாடுகளில் 18 தமிழ் இணைய மாநாடுகள் நடைபெற்றுள்ளன. இப்பொழுது TIC 2020 பத்தொன்பாவது மாநாடு இணையவழி நிகழவுள்ளது அறிய மிகுந்த மகிழ்ச்சி அளிக்கின்றது.

இந்த 23 ஆண்டுகளில் நாம் சாதித்தவை என்ன என்று எண்ணிப் பார்க்க வேண்டிய நேரம் இது. ஏராளமான தமிழ் இணையதளங்கள் பிறந்துள்ளன. பல்வேறு எழுத்துரு தரப்பாடுகளிலிருந்து இரண்டு எழுத்துருத்தரப்பாடுகள் - ஒருங்குறி தமிழ் (Unicode Tamil), அனைத்து எழுத்துருத்தரப்பாடு (TACE-16), ஆகிய இரண்டு தரப்பாடுகள் தமிழக அரசால் அரசின் தரப்பாடுகளாக ஏற்கப் பெற்றுள்ளோம். சொற்செயலிகள், தமிழ் எழுத்துரு அறி மென்மம் (Tamil OCR), பேச்சுத்தமிழை எழுத்துத்தமிழாக்கும் மென்மம், எழுத்துத்தமிழை பேச்சுத்தமிழாக்கும் மென்மம், தமிழ் இயல் மொழிச்செயலாக்கம் போன்ற பல்வேறு மென்பொருள்கள் உருவாக்கப் பெற்றுள்ளோம். உலகு தழுவி வாழும் தமிழ் மக்களும், தமிழில் ஈடுபாடு உள்ள மற்றையோரும், தமிழ் மொழியைக் கற்கவும், தமிழர் வரவாறு, கலை, இலக்கியம், பண்பாடு பற்றி அறிந்து கொள்ளவும், மழலைக்கல்வி முதல் பட்டப் படிப்பிற்கான பாடப் பொருள்களையும், மிகப்பெரிய தமிழ் மின்நூலகத்தையும் தன்னுட்கொண்டு தமிழ்ப்பணி ஆற்றி வரும் சிறந்ததொரு தமிழ் இணையப் பல்கலைக்கழகம் கிடைக்கப் பெற்றுள்ளோம.

ஆனால் இப்படைப்புகளெல்லாம் மக்களைப் போய்ச் சேர்ந்துள்ளனவா? தமிழை ஆட்சி மொழியாகவும் வழக்கு மொழியாகவும் கொண்டுள்ள நாடுகளின் அரசுக்கள் ஏற்று செயல் படுத்துகின்றனவா? இல்லையெனில் அதற்கு உத்தமம் என்ன செய்யவேண்டும்? எப்படிச் செயல்படவேண்டும்? என்பவற்றை இம்மாநாட்டில் கலந்தாய்ந்து முடிவு செய்யவேண்டிய நிலையில் உள்ளோம். . மேலும் இணையதள தமிழ் முகவரி வடிவமைக்கும் பணி இன்னும் நிறைவு பெறாமல் உள்ளது. இப்பணி நிறைவுபெற செய்ய வேண்டியவை பற்றிய முடிவெடுக்க வேண்டிய நிலையில் உள்ளோம்.

இணையப் பயன்பாட்டிற்கு ஒருங்குறி தமிழ் எழுத்துரு (Unicode Tamil) தரப்பாடு என்றும், பிற பயன்பாடுகளுக்கு அனைத்து எழுத்துரு (TACE-16) தரப்பாடு என்றும் தமிழக அரசு அறிவித்துள்ளது. எல்லாப் பன்பாடுகளுக்கும் அனைத்து எழுத்துரு (TACE-16) தரப்பாடே சிறந்தது என்பதை பல்வேறு ஆய்வுகள் வெளிப்படுத்தியுள்ளன. இந்த அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டின் பயன்பாடு கூடினால், ஒருங்குறி சேர்த்தியம் (Unicode consortium) ஒருங்குறி தளத்தில் 32-பிட்டு அமைப்பில் இந்த அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டினை சேர்க்க இசைவளித்துள்ளது. எனவே, அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டினை பல்வேறு பயன்பாடுகளில் செயல்படுத்தி அதன் பயன்பாட்டினை பெருக்குமாறு கணித்தமிழ் அன்பர்களையெல்லாம் அன்புடன் கேட்டுக் கொள்கின்றேன்.

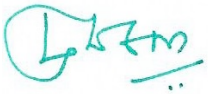
இன்று ஜப்பான், கொரியா போன்ற நாடுகளில் செயல்படும் கணிப்பொறிகளுக்கு ஆங்கிலம் தெரியாது. ஜப்பான், கொரிய மொழிகளில்

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

கொடுக்கப்படும் கட்டளைகளை மட்டுமே புரிந்நதுகொண்டு செயல்படுகின்றன. உலக மொழிகளின் தாய்மொழியாகிய தமிழ் மொழிக் கட்டளைகளால் இயங்கும் கணிப்பொறியை வடிவமைக்க இயலாதா என்ன? அனைத்து எழுத்துரு (TACE-16) தரப்பாடு செய்யப்படும்வரை இயலா நிலை இருந்தது. இப்பொழுது அத்தடை இல்லை. இனி அப்படியொரு கணிப்பொறியைக் காண்பது எப்போழ்து? Assembler போன்ற அமைப்புச் செயல்மொழி (System software) windows போன்ற இயக்க மென்பொருள் (Operating System) ஆகியவற்றை அனைத்து எழுத்துரு (TACE-16) தரப்பாட்டில் வடிவமைத்து ஒரு முழுமையான தமிழ்க்கணினியை படைக்க முடியும். இப்படியொரு முழுமையான தமிழ்க்கணினியை படைத்து வழங்குபவருக்கு உருபா ஒரு இலக்கம் பரிசு வழங்கப்படும் என்று மலேசியாவில் நடைபெற்ற 12-வது தமிழ் இணைய மாநாட்டின் போழ்தே அறிவித்திருந்தேன். ஆனால் இன்றுவரை அது நிகழவில்லை. விரைவில் அப்படியொரு முழுமையான தமிழ்க்கணினி பயன்பாட்டிற்க்கு வரவேண்டும் என்பது எனது பேரவா? என் கனவு நிறைவேறுமா?

பத்தொன்பாவது தமிழ் இணைய மாநாடு சிறப்பாக நடைபெற எனது உளங்கனிந்த வாழ்த்துக்களை தெரிவித்துக் கொள்கின்றேன். வாழ்க தமிழ்! வளர்க கணித்தமிழ்!

அன்புடன்,



(மு.பொன்னவைக்கோ)

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

பெரியார் பல்கலைக்கழகம்

(தேசிய தரமதிப்பீட்டுக்குழுவின் மறுமதிப்பீட்டில் 'A' தரச்சான்றிதழ் பெற்றது)

பெரியார் பல்கலை நகர்

சேலம் - 636 011, தமிழ்நாடு, இந்தியா

பேராசிரியர் பொ. குழந்தைவேல்

துணைவேந்தர்

நாள்: 04-12-2020

வாழ்த்துரை

உலகத்தகவல் தொழில்நுட்ப மன்றம் (INFITT) என்றழைக்கப்படும் உத்தமம் அமைப்பானது இணையத்தில் தமிழ் சார்ந்த தகவல் தொழில்நுட்பங்களை வளர்தெடுக்க பல முயற்சிகளில் ஈடுபட்டு வருகிறது. உலகெங்கிலும் உள்ள தமிழ் தகவல் நுட்ப ஆராய்ச்சியாளர்களையும், சர்வதேச அளவில் இணையத்தமிழ் ஆய்வுகளையும் ஒருங்கிணைத்து தமிழையும் தமிழர்களையும் அடுத்தகட்ட வளர்ச்சியை நோக்கி நகர்த்திச் செல்வது இதன் நோக்கமாகும்.

கணினித்தமிழை வளர்தெடுக்க முழு முனைப்புடன் ஈடுபட்டு வரும் இந்த உத்தமம் அமைப்பின் ஏற்பாட்டில் பல முன்னனி நாடுகளின் ஆதரவுடனும், பன்னாட்டு நிறுவனங்கள் மற்றும் அறிவார்ந்த தொழில்நுட்ப ஆய்வாளர்களின் ஒத்துழைப்போடும் உலகத்தமிழ் இணைய மாநாடுகள் அரங்கேறியுள்ளன.அதற்கு முதலில் எனது வாழ்த்துகளைத் தெரிவித்துக் கொள்கிறேன்.

1997 ஆம் ஆண்டுத் தொடங்கிய உத்தமத்தின் தன்னிகரற்ற இணையத் தமிழ்ப் பயணம், தற்போது 2020 ஆம் ஆண்டில் காலடி வைத்துள்ளதைப் பெருமைக்குரியதாக நினைத்து மகிழ்ச்சியடைகிறேன்.

தற்போது 19வது உலகத் தமிழ் இணைய மாநாடு (2020) உத்தமத்தின் வளர்ச்சி சார்ந்த ஆதரவோடு, இலங்கை – மெராட்டுவப் பல்கலைக்கழகத்தின் தேசிய மொழிகள் ஆய்வு மையமும் - சேலம் பெரியார் பல்கலைக்கழகமும் இணைந்துநடத்த உள்ளது. இம்மாநாடு தமிழ்க் கணினி ஆய்வு – கணினி வழி தமிழ் ஆய்வு - இவற்றின் வளர்ச்சியையும் சவால்களையும் ஆய்வுக்களமாகக் கொண்டு செயல்பட உள்ளது.

இம்மாநாடு சில சூழல்களைக் கருத்தில் கொண்டு, இணைய வழியில் நடைபெற்றாலும், முதன்முதலில் நடைபெறும் மெய்நிகர் மாநாடு என்ற வரலாற்றுச் சிறப்பினைப் பெறவுள்ளது என்பதைப் பெருமையோடு கூறுவதற்கு என்றும் கடமைப்பட்டுள்ளேன்.

மெய்நிகர் மாநாடாக நடைபெறுவதன் வழி கணினி வழி மொழிபெயர்ப்பு, இணையம் வழியிலான கற்றல் - கற்பித்தல் முறைகள், கணினி வழியிலான தமிழ் மொழி பகுப்பாய்வுகள், கணினியோடு செயல்படக்கூடிய தமிழ் உரைவழிப்பட்ட பேச்சு, தமிழாக்கப்பட்ட பயன்பாட்டு நிரலிகள்.... போன்றவற்றில் வளர்ச்சி நிலை அடைந்து இணையத் தமிழ் தொடர்பான சிக்கல்களைத் தீர்த்து எதிர்காலப் பிரச்சனைகளை எளிமையாக்கும் என நம்புகிறேன்.

உதாரணமாகச் சொல்ல வேண்டுமென்றால், ஏட்டுக் கல்வியைத் தாண்டி, சாதாரண மக்களும் பிற ஆர்வலர்களும் உலக அறிவை எளிமையாய்ப் பெறுவதற்கு வழிவகை செய்யும். அவரவர் திறமைக்கு ஏற்ப, தானாகவே கல்வித் திட்டங்களில் இணைந்து சுய மற்றும் பொது அறிவை வளர்த்துக்கொள்ள முடியும்.

உத்தமம் என்னும் உலகத் தகவல் தொழில்நுட்பமன்றத்தினருக்கும் இணையத்தின் வழியிலான தமிழ் வளர்ச்சிக்கு இணைந்துள்ள பிற அமைப்பு மற்றும் அமைப்புச் சாரா தன்னார்வலர்களுக்கும், ஆய்வுக் கட்டுரை வழங்கி இணையத் தமிழ்ச் சிந்தனையை மேலும் கூர்மைப்படுத்தும் கட்டுரையாளர்களுக்கும், பல நாடுகளைச் சார்ந்த கணிப்பொறி வல்லுநர்களுக்கும், பிற துறைச் சார்ந்த வல்லுநர்களுக்கும் என் நெஞ்சம் நிறைந்த வாழ்த்துகளைத் தெரிவித்துப் பெரு மகிழ்ச்சியடைகிறேன். வாழ்க தமிழ்! வளர்க இணையத் தமிழ்....!!!

பொ. குழந்தைவேல்.

[பொ. குழந்தைவேல்]

தொலைபேசி எண் 0427 - 2345161
தொலை நகலி எண் 0427 - 2345565
மின்னஞ்சல் vcperiyar@gmail.com
வலைதளம் www.periyaruniversity.ac.in

எழுதாணித் தமிழ், கல்வெட்டுத் தமிழ், பேனாத் தமிழ், அச்சுத்தமிழ் என்று தமிழ் மாறிக்கொண்டே வருகிறது. இன்று கணினித் தமிழ், இது கன்னித் தமிழின் புதிய பரிமாணம். மற்றவையெல்லாம் தமிழை வெளிப்படுத்த வந்தவை. கணினித் தமிழ் வெளிப்பாட்டோடு தமிழின் பயன்பாட்டையும் மாற்றவல்லது. இந்த மாற்றத்தை முன்னெடுத்துச் செல்வது கணினி வல்லுநர்களின், மொழி வல்லுநர்களின் கூட்டு முயற்சி. இந்த முயற்சியின் சாத்தியங்களையும் சாதனைகளையும் காட்டுவது இந்த இணைய மாநாடு. தமிழின் எதிர்காலத்துக்குப் பல்லக்குத் தூக்கும் இந்த மாநாடு வெற்றிபெற வாழ்த்துகள். இந்த வெற்றி தமிழ் பற்றிய நம்பிக்கை பெறும் வெற்றி.

பேராசிரியர் இ. அண்ணாமலை
சிகாகோ பல்கலைக்கழகம்

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

இடுக்கண் வருங்கால் முயல்க!

வணக்கம். பத்தொன்பதாவது தமிழ் இணைய மாநாட்டின் நிகழ்ச்சிகள் குழுவுக்குத் தலைமையேற்றுப் பணிபுரிய வாய்ப்பளித்த உத்தமம் நிறுவனத்துக்கு நன்றி. மனித நேயம் இதுவரை கண்டிராத இடுக்கண் நிறைந்த இக்காலகட்டத்திலும் நாம் முயன்றோம். பேராசிரியர் மு. அனந்தகிருஷ்ணன் மற்றும் பேராசிரியர் பொன்னவைக்கோ ஆகியோர் தொடர்ந்து அளித்துவரும் ஊக்கமும் வழிகாட்டலும் இம்மாநாட்டின் வழித் தமிழ்க் கணினி ஆய்வு தொடர்ந்து நடைபெற ஏதுவாக இருக்கிறது என்றால் அது மிகையாகாது. உத்தமம் மாநாட்டின் வழித் தங்களின் ஆய்வுகளை அரங்கேற்றித் தமிழுலகுக்குத் தமிழ்க் கணினியின் வளர்ச்சியைக் காட்டிவரும் கணினித் தமிழ் ஆய்வாளர்களின் முயற்சியைத் தலைவணங்குவோம். தமிழைப் பைத்தான் போன்ற நிரலிகளின் வழி ஆய்ந்தறியும் இக்காலகட்டம் எழுத்தாணி கொண்டு ஓலைக்கீரலின் வழி ஆய்ந்தறிந்த காலகட்டத்திலிருந்து முற்றிலும் வேறுபட்டதல்ல. இருவேறு ஆய்வாளர் பரம்பரைகள் ஒருமித்த தமிழ் மணம் வீசும் ஆய்வுமனம் கொண்டோர். தமிழ்ப் பிராமி, தமிழி, வட்டெழுத்து எனத் தொடங்கிய பயணத்திலிருந்து இன்று ஒருங்குறிக்குப் பயணித்த காலந்தான் வியக்கத்தக்க காலமல்லவோ! தமிழ் எழுத்திலிருந்து உருபனுக்கும், உருபிலிருந்து இலக்கணத்துக்கும் என தமிழைப் பல்வேறு கோணங்களில் காணும் ஆய்வுக்கட்டுரைகளை இக்கட்டுரைத் தொகுப்பில் காணலாம். இவ்வாய்வாளர்கள் இந்த இடுக்கண் காலத்தில் முயன்ற படைப்புகள் இவை. இவர்களது முயற்சிகள் தொடரட்டும். தண்கதிர் மதியம் போலவும் ஒண்கதிர் ஞாயிறு போலவும் மன்னிய பெரும நீவிர் நிலமிசையானே!

வாசு அரங்கநாதன்
மாநாட்டு நிகழ்ச்சிகள் குழுத் தலைவர்
பென்சில்வேனியாப் பல்கலைக்கழகம்.

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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 *typeface*. For the sake of avoiding jargon, *font* is used instead**

2 A Brief Overview of Tamil Scripts

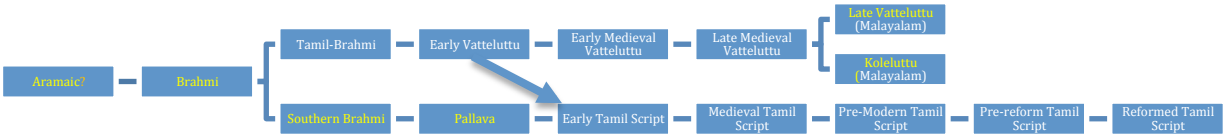
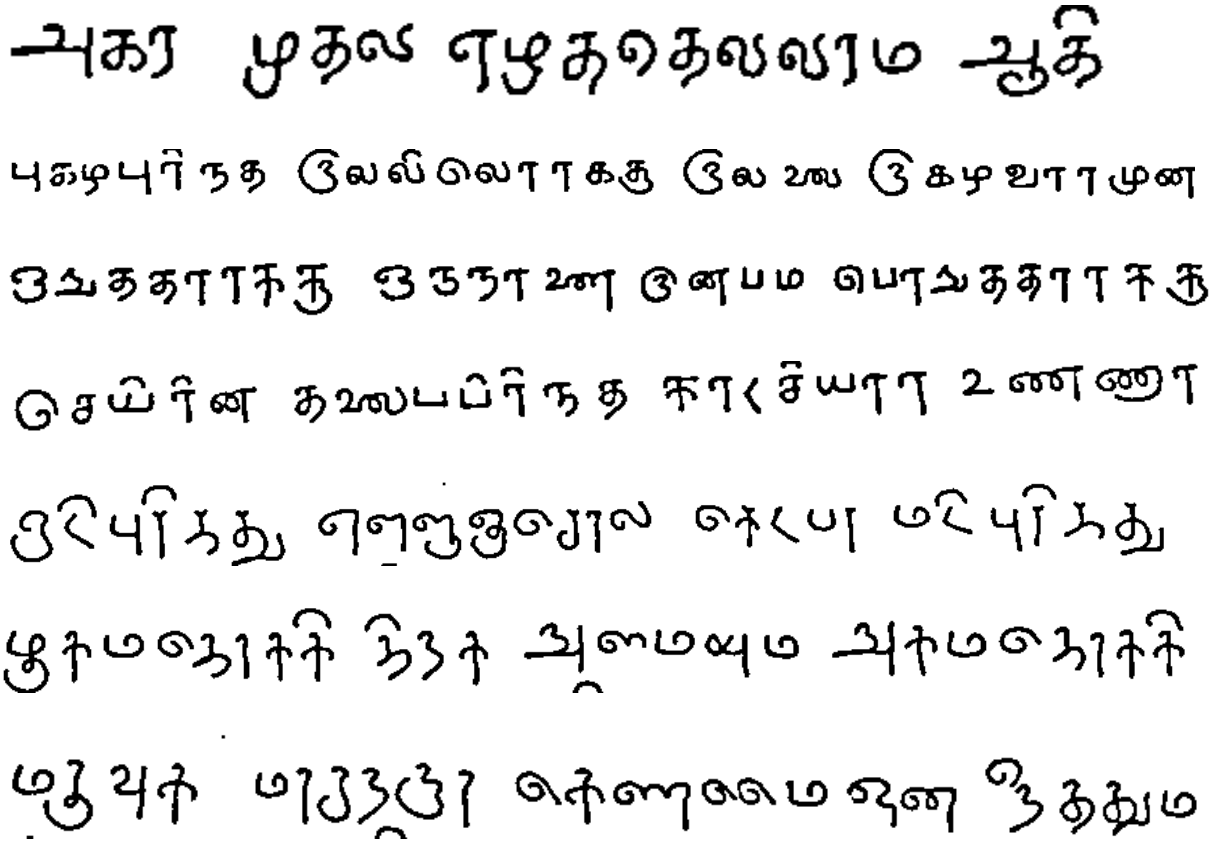


Fig 1: The evolution of Tamil scripts

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*[2], 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] வட்டெழுத்து

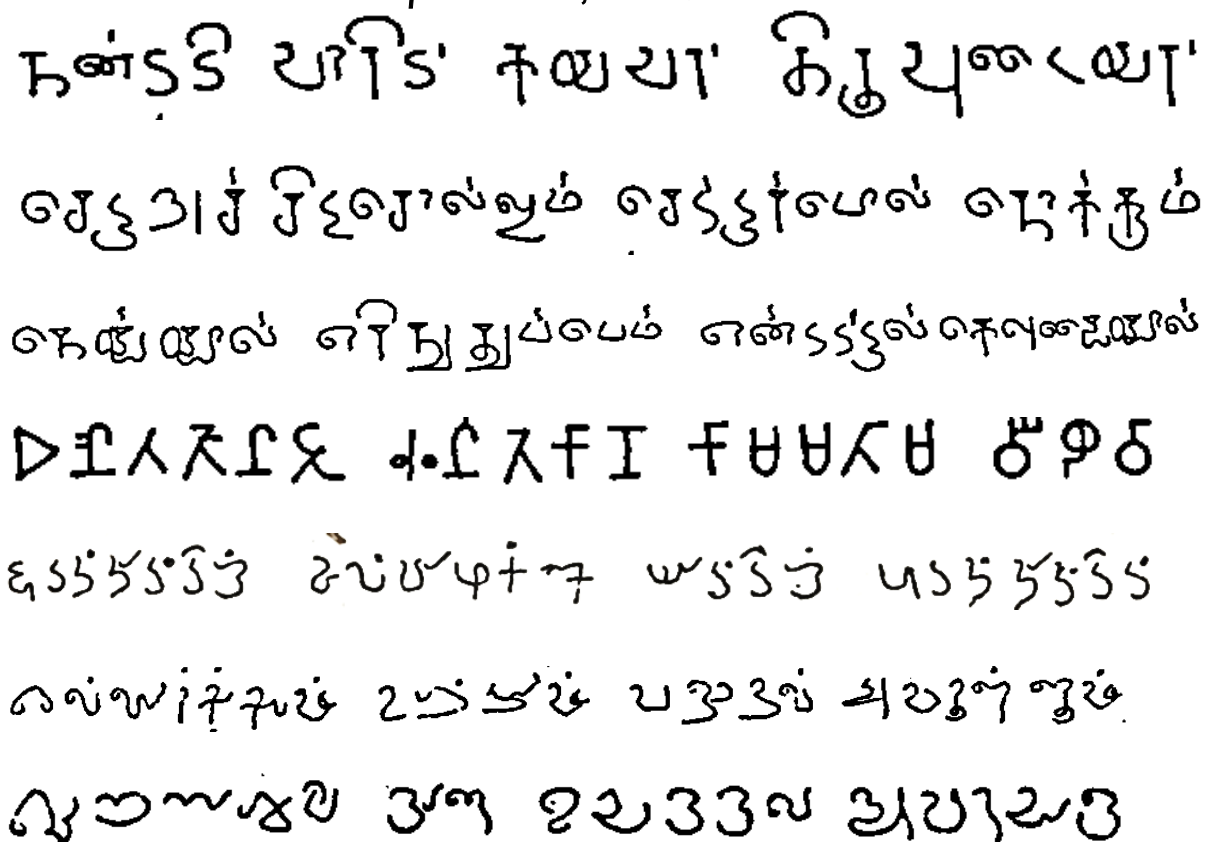


Fig 2: Samples of historical Tamil scripts from Siromoney et al. (1980) and Siromoney et al. (1981)

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 *cērā* 𑀘𑁂𑀭𑀸 is actually an interpretation of the original ambiguous sequence 𑀘𑁂𑀭𑁦 which in all probability could also be read as cērar 𑀘𑁂𑀭𑀭𑁆. 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*.

Fig 3: Jambai Inscription (Mahadevan, 2003)

Rendering and archiving it with the apparent reading as:

ஸதியபுதோநேடுமாநஅஞிசிஈததபாளி[3]

or the intended reading as:

ஸதியபுதோநெடுமாந்அஞ்சிஈத்தப(ள்)ளி[4]

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

𑀲𑀢𑀺𑀬𑀧𑀼𑀢𑁄𑀦𑁂𑀝𑀼𑀫𑀸𑀦𑀅𑀜𑁆𑀘𑀺𑀈𑀢𑁆𑀢𑀧(𑀴𑁆)𑀴𑀺

(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 𑀶𑁆𑀓𑁆𑀫𑁆𑀶𑀸𑀯𑁆 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] satiyaputōnēṭumānañciītatapāḷi

[4] satiyaputōneṭumānañciīttapa(ḷ)ḷi

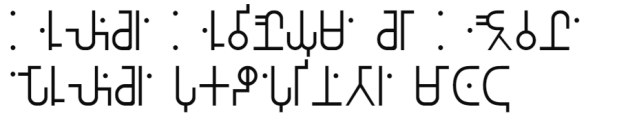
For instance, in Tamil-Brahmi 𑀉 */u/* and 𑀭 */ru/* are similar and in *Vatteluttu* 𑜆 /ṭa/, 𑜉 */pa/* & 𑜊 */va/* are similar, even though the characters look visually distinct in the modern script.

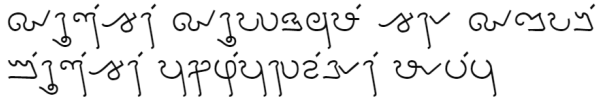
Fig 4: 𑀉 /u/ and 𑀭 /ru/ in Tamil-Brahmi

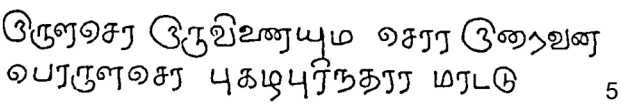
Fig 5: 𑜆 /ṭa/, 𑜉 /pa/ & 𑜊 /va/in Vatteluttu

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.







இருளசொ இருவினையும சொா இறைவன
பொருளசொ புகழபுரிநதா மாடடு

இருள்சேர் இருவினையும் சேரா இறைவன்
பொருள்சேர் புகழ்புரிந்தார் மாட்டு[6]

[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:

> பத்துரதன் புத்திரனின் மித்திரனின் சத்துருவின்
> பத்தினியின் காலைவாங்கி தேய்[7]
>
> *Rub [against it] after removing the leg of the wife (tārai தாரை) of the enemy (Vālī) of the friend (Sugrīva) of the son (Rāma) of Ten-rathas (Daśaratha).*

If the *kāl ◌ா is to be removed from தாரை*, it becomes *tarai தரை* 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*[8]. The 11th century Tamil grammatical compendium *Yāpparuṅkala Virutti*[9] describes it as follows (Pillai, 1998):

> பிந்துமதி என்பது எல்லா எழுத்தும் புள்ளியுடை யனவே வருவது[10]
> *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.

> நெய்கொண்டெ னெற்கொண்டெ னெட்கொண்டென் கொட்கொண்டென்
> செய்கொண்டென் செம்பொன்கொண் டென்[11]

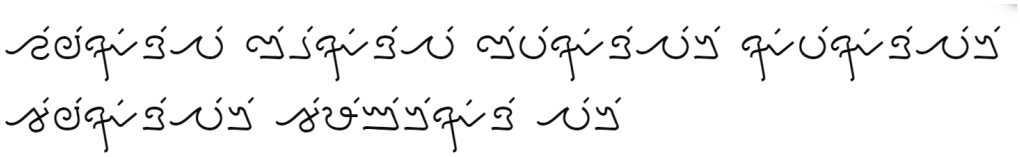
---

[6] **iruḷcēr iruviṉaiyum cērā iṟaivaṉ**
poruḷcēr pukaḻpurintār māṭṭu
[7] **patturataṉ puttiraṉiṉ mittiraṉiṉ catturuviṉ**
pattiṉiyiṉ kālaivāṅki tēy
[8] பிந்துமதி
[9] யாப்பருங்கல விருத்தி
[10] **pintumati eṉpatu ellā eḻuttum puḷḷiyuṭai yaṉavē varuvatu**

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.

நெய்கொண்டெ் னெற்கொண்டெ் னெட்கொண்டென் கொட்கொண்டென்
செய்கொண்டென் செம்பொன்கொண்ட டென்

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 ெ and ா do not fuse.



Another instance would be the reference *ஙகர வெல்கொடியான்*[12] appearing in the *Tevāram* verse 97:16. This is glossed as *இடபக்கொடியான்*[13] *(bull-flagged-one)* and further expanded as *படுத்திருக்கும் வடிவில் இடபம் ங போன்றிருத்தலின்*[14] *(because the figure of a bull lying resembles [the letter] ங /*ṅa*/) (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: ங /*ṅa*/ 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] **neykoṇṭe ṉeṟkoṇṭe ṉeṭkoṇṭeṉ koṭkoṇṭeṉ ceykoṇṭeṉ cemboṉkoṇ ṭeṉ**
[12] **ṅakara velkoṭiyāṉ**
[13] **iṭapakkoṭiyāṉ**
[14] **paṭuttirukkum vaṭivil iṭapam ṅa pōṉṟiruttaliṉ**

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 சா◌ா could be read as any of the following: cerā செரா, cērā சேரா, cor செர், cōr சேர், cerar செரர், cērar சேரர் 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 retreival/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?

ஆ	அம்
கூ	க் க
ற	க் கு
ற்	க் கும்
கூ	கம்
யூ	யம், யும்
ர	ார்
ர	ார, ரா
ரூ	ராம, ாரம்
ளு	ளம்
ணூ	ணம்
னு	னம்

Fig 7: Some ligatures ocurring in Tamil manuscripts (Srinivas et al., 2001)

The overaching 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 *கடைவிரித்தும் கொள்வாரில்லை*[15] *(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 koḷvārillai**

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 𑀬 /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

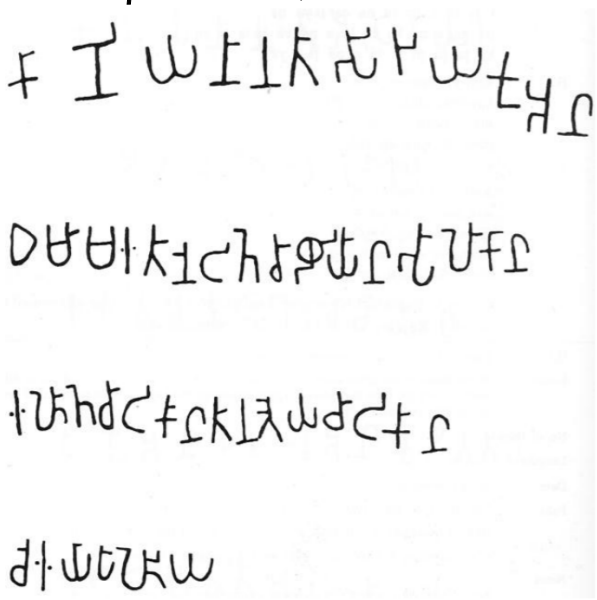
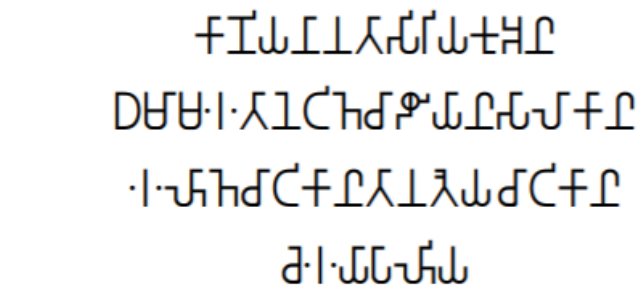


Fig 9: Mangulam Inscription (Mahadevan, 2003)



Actual Reading

காணியநாநதாஸிரியகுஅன
த⁴மாமஈதாநேடிஞ்சாழியானஸாலாகான
ஈளாஞ்சாடிகானதாநதையசாடிகான
சேஈயாபாளிய

kāṇiyanānatāsiriyakuaṉa
dhamāmaītānēṭiñacāḻiyāṉasālākāṉ
īḷāñacāṭikānatānataiyacāṭikāṉa
cēīyāpāḷiya

(Tamil Brahmi I - Notation)

Intended Reading

கணிய்-நந்தஸிரிய்-குஅன்
த⁴(ம்)மம்-ஈதா-நெடிஞ்சழியன்-ஸாலகன்
இளஞ்சடிகன்-தந்தைய்-சடிகன்
சேஇய-ப(ள்)ளிய்

kaṇiy-nantasiriy-kuaṉ
dha(m)mam-ītā-neṭiñcaḻiyaṉ-sālakaṉ
iḷañcaṭikaṉ-tantaiy-caṭikaṉ
cēiya-pa(ḷ)ḷiy

Fig 10: Digitization from Adinatha Tamil-Brahmi Font Manual (Rajan, 2012)

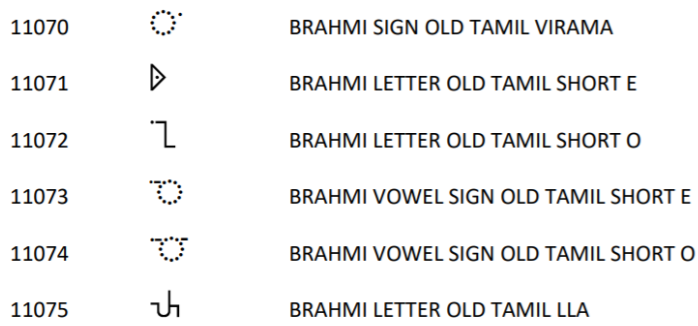


Fig 11: New Tamil-specific additions to Tamil-Brahmi (Rajan & Sharma, 2019)

The discussion of the complete design process behind the Adinatha Tamil-Brahmi font can be seen in *Rajan & Sankar (2020)*.

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*[16] 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**

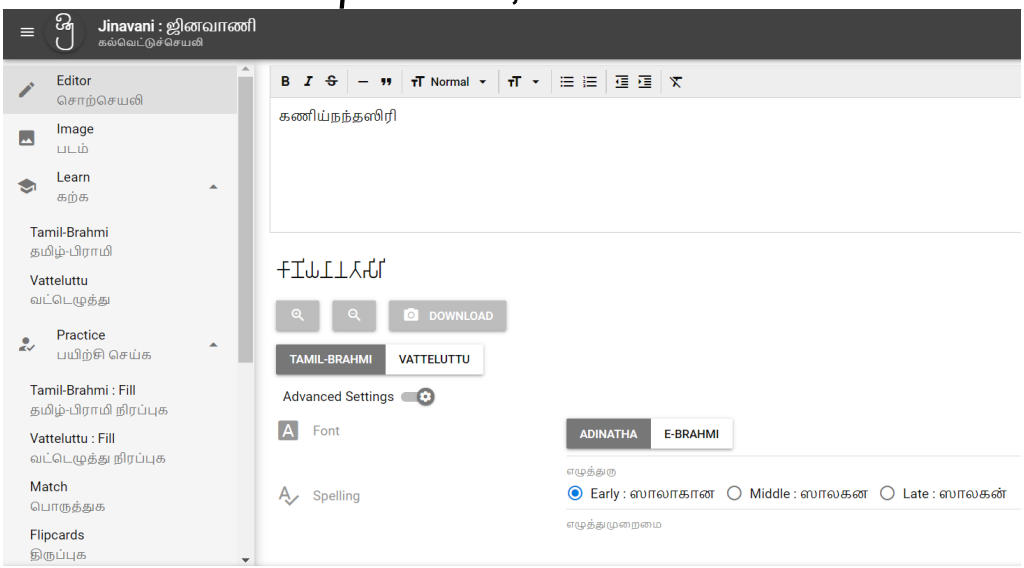


Fig 12: Jinavani Home Page

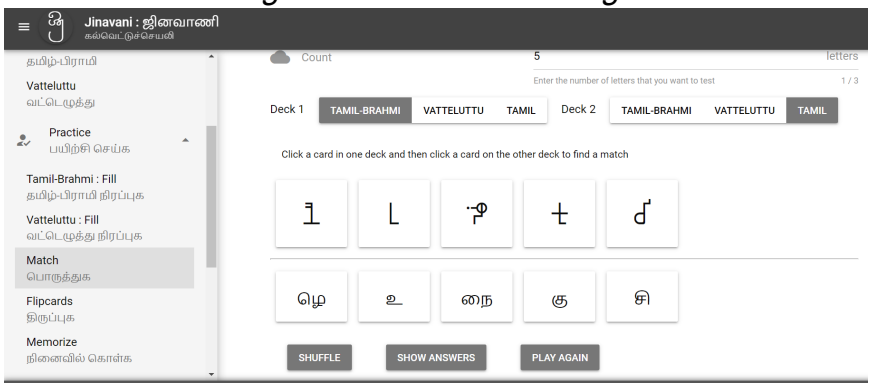


Fig 13: Jinavani Gamified Learning 1

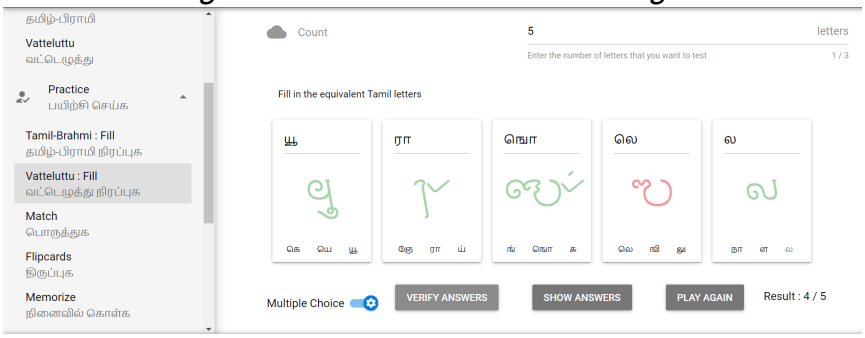


Fig 14: Jinavani Gamified Learning 2

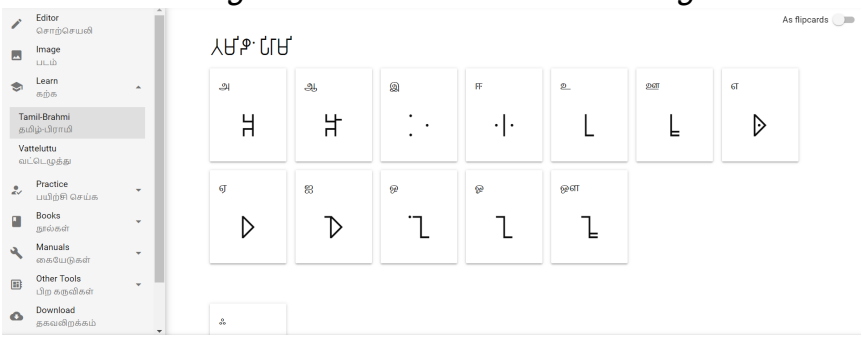


Fig 15: Jinavani Learning Cards

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 *Ṅakaravelkoṭi.* 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

1. Mahadevan, I (2003): *Early Tamil epigraphy: From the Earliest Times to the Sixth Century A.D*. Chennai, India. Cre-A. (Harvard Oriental Series, Volume sixty-two.)
2. Piḷḷai, V. (1998). Yāpparuṅkalam - Paḷaiya Viruttiyuṭaṉ. Ulakat Tamiḻ Ārāycci Niṟuvaṉam, Ceṉṉai.
3. Rajan, V & Sankar, U. (2020). *Contemporizing a Historic Script: The Development of Adinatha Tamil Brahmi Typeface and its Community Impact*. Atypi 2020. Association Typographique Internationale. https://events.bizzabo.com/254216/agenda/session/418615
4. Rajan, V (2012). Adinatha Tamil-Brahmi Font Manual. http://virtualvinodh.com/pdfs/Adinatha_Tamil_Brahmi_Manual.pdf
5. Rajan, V. & Sharma, S. (2019). L2/19-402: *Proposal to Encode 6 Characters in the Brahmi Block*. L2 Registry, Unicode Consortium.
6. Sacchidanandan, M., -. Paṉṉiru Tirumuṟai Pāṭṭum Poruḷum. [online] Thevaaram.org.

[Accessed 6 December 2020].
Available at:
<http://www.thevaaram.org/thirumurai_1/songview.php?thiru=5&Song_idField=50970&padhi=097&startLimit=16&limitPerPage=1&sortBy=&sortOrder=DESC>
(See also: https://www.ifpindia.org/digitaldb/site/digital_tevaram/INDEX.HTM)

7. Siromoney, G., Govindaraaju, S. & Chandrasekaran, M. (1980). *Tirukkural in Ancient Scripts*. Tambaram. Department of Statistics, Madras Christian College.
8. Siromoney, G., Govindaraaju, S., Chandrasekaran, M. & Chandrasekaran, S. (1981). *Vaṭṭe̲luttil Tirukku̲ra̲ḷ*. Tambaram, Department of Statistics, Madras Christian College.
9. Srinivas, M.D., Paramasivam, T.G. & Pushkala, T. (2001). *Thirupporur and Vadakkuppattu: Eighteenth Century Locality Accounts*. Centre for Policy Studies, Chennai.
10. Visalakshy, P. (2003). The Grantha Script. Thiruvananthapuram: Dravidian Languages Association.

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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

Louis Isack Kumar,

Bartley Secondary School, Singapore

Abstract:-

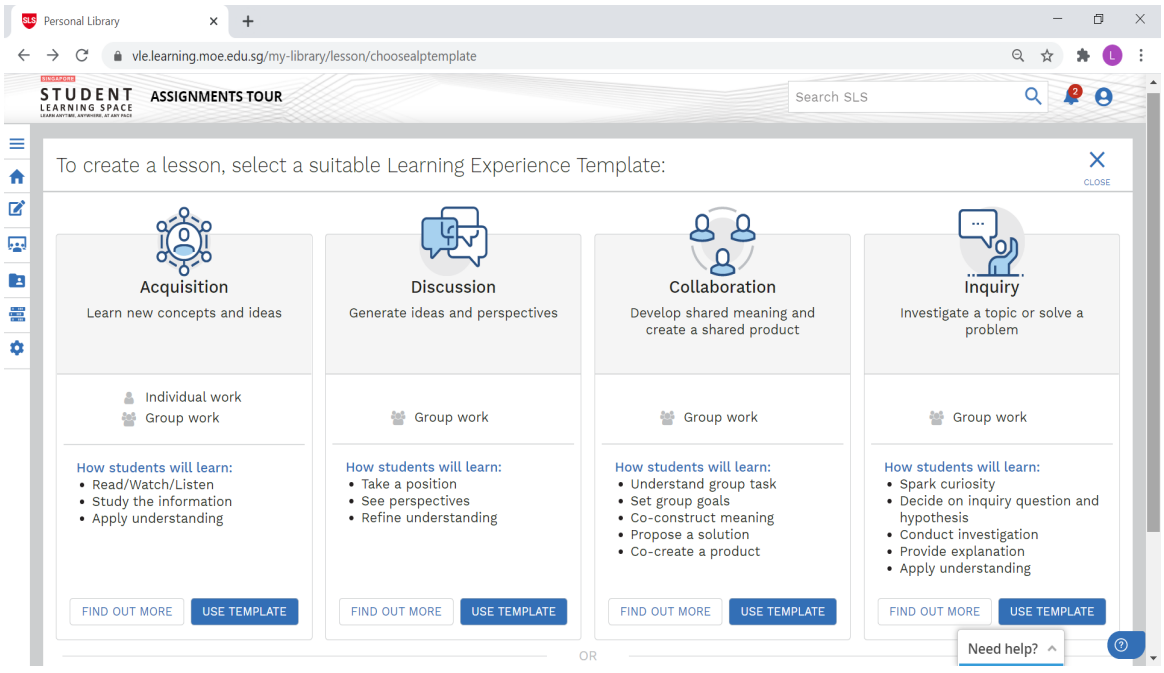
This paper aims to provide an overview of Student Learning Space (SLS), an online learning portal used in all schools in Singapore. This portal aims to support students in self-directed learning at school and beyond the classroom. This paper discusses how SLS and Visible Thinking Routines are implemented 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.

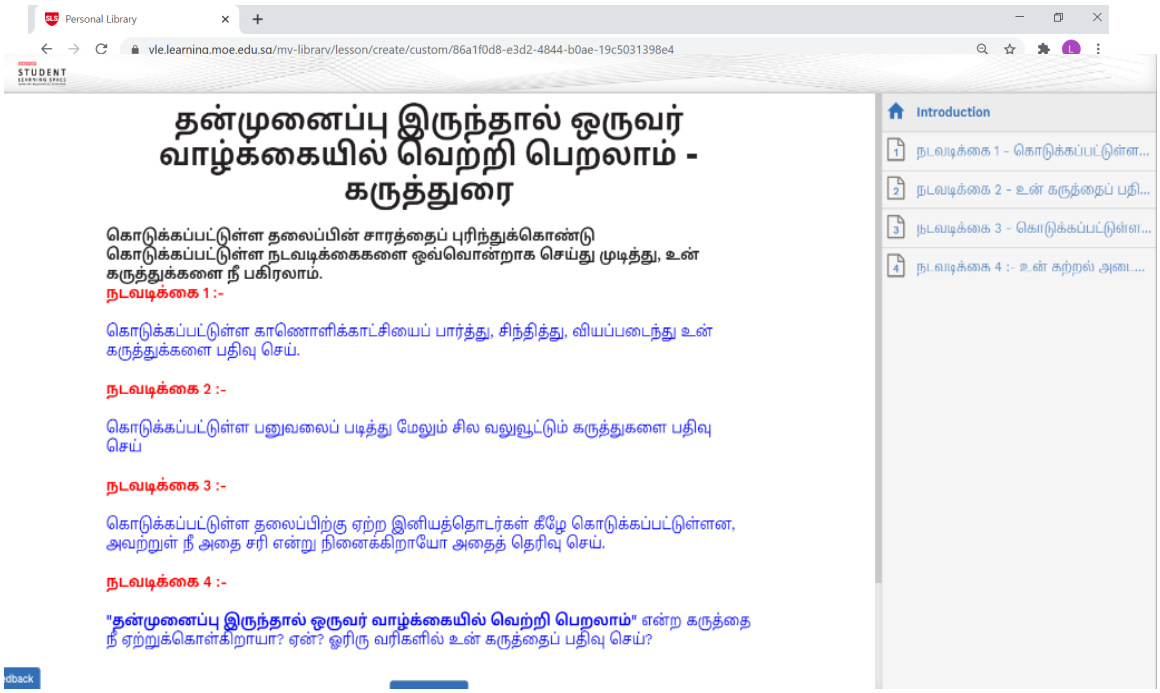
Making thinking visible must be at the heart of improving teaching and learning Tamil as a second language in classrooms. With the current Singapore teaching methods, students' thinking processes is often used in classroom teaching and are well enhanced and developed. Through the implementation of thinking 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

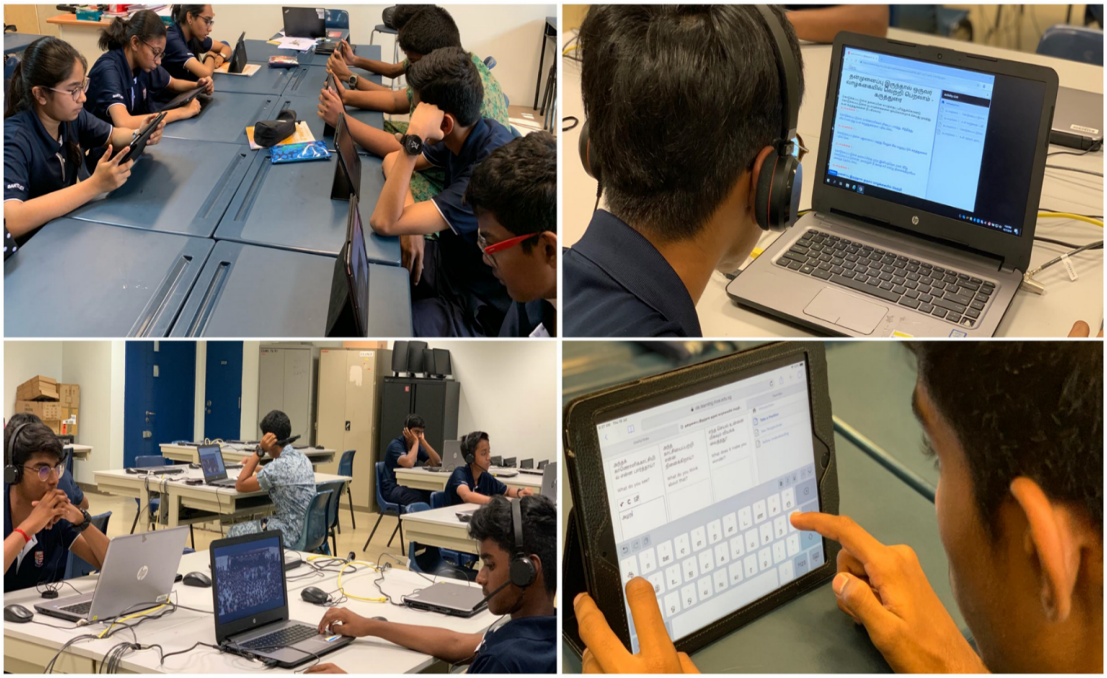
thinking and structure classroom discussion. Begin by thinking about the type of thinking you want your students to do and try to identify a thinking routine that will help bring that to the foreground.

Interactive Thinking Tools in SLS:-

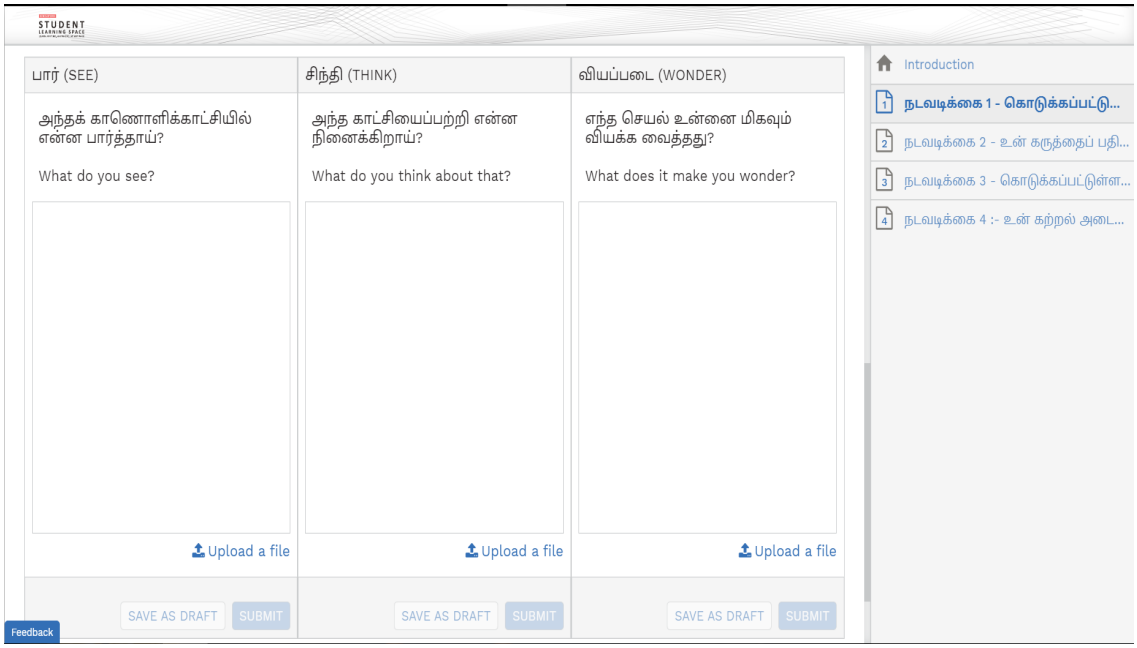
Thinking routines are intended to “support and structure students’ thinking. The steps of the routine act as natural scaffolds that can lead students’ thinking to higher and more sophisticated levels” (Ritchhart, Morrison, & Church, 2011, p. 47). When teacher plan activities using SLS portal and multimedia 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.



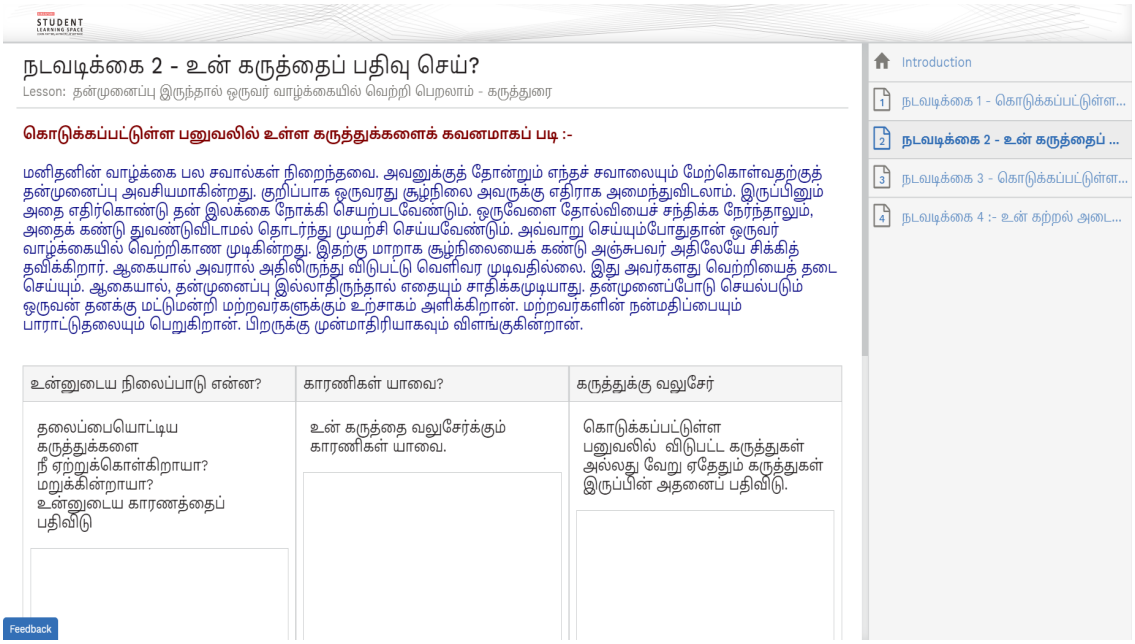
Expository writing Task in SLS portal using Thinking Routines



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

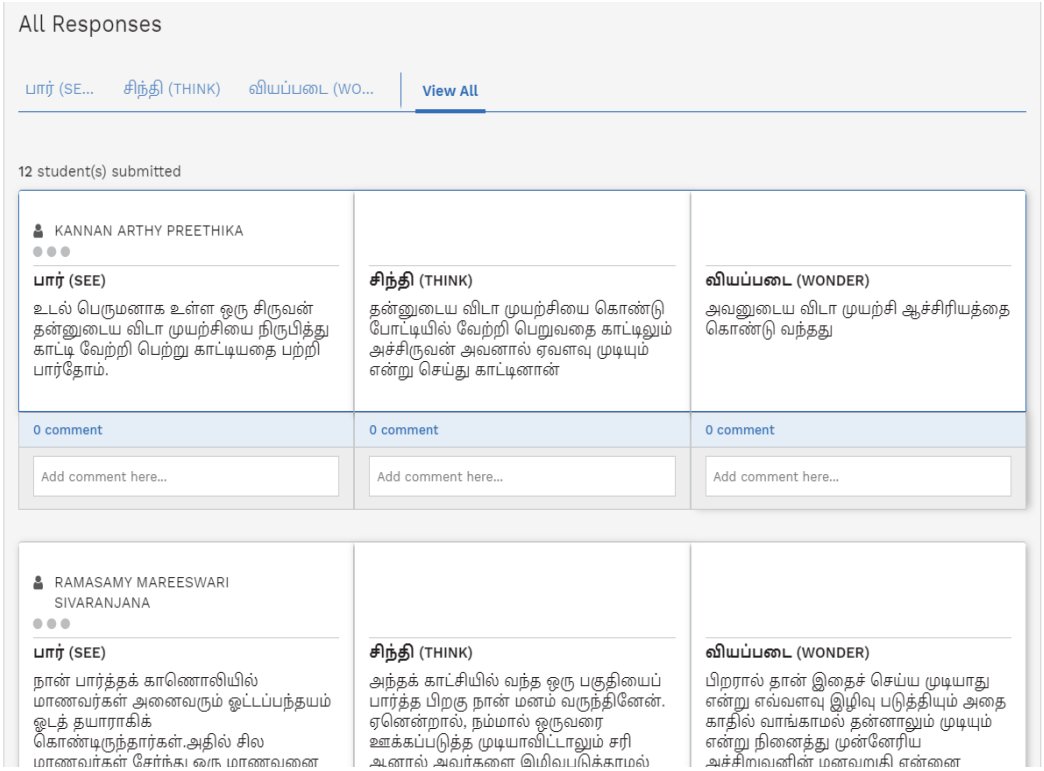


providing reasons. This is something that does not come naturally for students, and they need guidance in generating and selecting ideas during the pre-writing stage. With students mapping their ideas, they are able to demonstrate the connections between ideas and how these develop. As teachers employ 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.

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

Online teaching and learning during COVID-19 pandemic period: The Mauritian experience.

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

smartphone 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 Dvision (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

Zone: **Subject:**

School: **Name of Educator:**

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.
Feedback on Teaching and Learning

Date	Grade	Online platform/tools and other educational resources used.	Number of pupils on roll	Number of pupils attending the session	Topic	Remarks

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:

Term	Date	No. of weeks
1st	Grade 1/Grade 7 Friday 10 January to Friday 3rd April 2020 All Grades (7 -13) Monday 13 January to Friday 3 April 2020	12
	School Holidays (2 weeks)	
2nd	Monday 20 April to Friday 17 July 2020	13

	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)	12
	Total	**37**

Timing of Assessments and Examinations Prior to COVID-19

Grade 5 Modular Assessment	November 2020
Grade 6 Modular Assessment	August 2020
Grade 6 -Primary School Achievement Certificate (PSAC) Assessment	November 2020
Grade 9 -National Certificate in Education (NCE)Assessment	October/ November 2020
School Certificate (SC)/Higher School Certificate (HSC) Exams Cambridge International Examinations	September/October/ November 2020

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

Term	Date	No. of weeks
1st	Grade 1/Grade 7 Friday 10 January 2020 to Wednesday 18 March 2020	10

	Easter holidays 23 March to 5 April 2020 (2weeks) and Covid-19 Break	
TV broadcast lessons (Grade 1 to 9) and Online teaching and learning (Grade 10 to 13)	5 April 2020 to 15 June 2020	11
2nd	Wednesday 01 July 2020 to Friday 16 October 2020 Mid- term break (1 week) Monday 19 October 2020 to Friday 23 October 2020 Monday 26 October – Friday 27 November 2020	22
	School holidays (5 weeks)	
3rd	Thursday 7 January 2021 to Friday 26 March 2021	12
	Total	**44 + 11 weeks online teaching**

Timing of Assessments and Examinations

Post COVID-19- 2020/2021

Grade 5 Modular Assessment	March 2021
Grade 6 Modular Assessment	Week 1 of Dec 2020
Grade 6 Primary School Achievement Certificate (PSAC) Assessment	March 2021

Grade 9 -National Certificate in Education (NCE)Assessment	April 2021
School Certificate (SC)/Higher School Certificate (HSC) Exams Cambridge International Examinations	April/ May/June 2021

School calendar 2021/2022

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

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

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

Chauhan, Sumedha. "A meta-analysis of the impact of technology on learning effectiveness of elementary students." *Computers & Education*, vol. 105, Feb. 2017, pp. 14-30, www.sciencedirect.com/science/article/abs/pii/S0360131516302172. Accessed 24 Nov. 2020.

"Covid-19 et confinement - Education : des cours à la télévision et en ligne destinés aux élèves." *Defimedia.info*, 1 Apr. 2020, defimedia.info/covid-19-et-confinement-education-des-cours-la-television-et-en-ligne-destines-aux-eleves. Accessed 24 Nov. 2020.

The COVID-19 pandemic has changed education forever. This is how. 2020, www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/. Accessed 3 Dec. 2020.

Dhawan, Shivangi. "Online Learning: A Panacea in the Time of COVID-19 Crisis." *Journal of Educational Technology Systems*, vol. 49, no. 1, June 2020, journals.sagepub.com/doi/full/10.1177/0047239520934018. Accessed 24 Nov. 2020.

Government of Mauritius. "Covid-19: Ensuring teaching and learning continuity through digital education is a must, says VPM." *Government Information Service*, 14 May 2020, www.govmu.org/English/News/Pages/Covid-19-Ensuring-teaching-and-learning-continuity-through-digital-education-is-a-must,-says-VPM.aspx. Accessed 24 Nov. 2020.

Jaulim. "Ariane Navarre-Marie : « Ce projet de loi vient reprendre les droits acquis des travailleurs » 14 mai 2020." *Defimedia*. *Defimedia.info*,

defimedia.info/ariane-navarre-marie-ce-projet-de-loi-vient-reprendre-les-droits-acquis-des-travailleurs. Accessed 24 Nov. 2020.

La Rédaction. "Education: managing the crisis." *L'express*, www.lexpress.mu/article/373540/education-managing-crisis. Accessed 24 Nov. 2020.

Learning Channels MBC Broadcast. 2020, education.govmu.org/Pages/Main/MBC-Channels.aspx. Accessed 3 Dec. 2020.

Mauritius Institute of Education. 2018, portal.mie.ac.mu/. Accessed 3 Dec. 2020.

Online Resources. 2020, education.govmu.org/Pages/Main/Online-Resources.aspx. Accessed 3 Dec. 2020.

Oxford University launches world's first COVID-19 government response tracker. 25 Mar. 2020, www.bsg.ox.ac.uk/sites/default/files/2020-03/Oxford-Covid-19-Government-response-tracker-press-release%20.pdf. Accessed 23 Nov. 2020.

Ramessur-Bhoyroo, Preity. "Covid-19/Confinement- Education : voici les détails des cours en ligne et à la télé." *Defimedia*, 1 Apr. 2020. *Defimedia.info*, defimedia.info/covid-19confinement-education-voici-les-details-des-cours-en-ligne-et-la-tele. Accessed 24 Nov. 2020.

Student Support Programme. 2020, ssp.moemu.org/eresources.php. Accessed 3 Dec. 2020.

"'Une éducation à deux vitesses', c'est en ces termes qu'Ajay Gunness a fait allusion aux élèves." *InfoMoris*, 26 Sept. 2020, infomoris.com/top-tv/une-education-a-deux-vitesses-cest-en-ces-termes-quajay-gunness-a-fait-allusion-aux-eleves/. Accessed 24 Nov. 2020.

Favale, T., Soro, F., Trevisan, M., Drago, I., & Mellia, M. 2020. Campus traffic and e-Learning during COVID-19 pandemic. Computer Networks, 176, 107290. Accessed 24 Nov. 2020.

Martin, A. 2020. How to optimize online learning in the age of coronavirus (COVID-19): A 5-point guide for educators. https://www.researchgate.net/publication/339944395_

20 Journal of Educational Technology Systems 49(1) How_to_Optimize_Online_Learning_in_the_Age_of_Coronavirus_COVID-19_A_5-Point_Guide_for_Educators. Accessed 24 November 2020

Seville, E., Hawker, C., & Lyttle, J. 2012. Resilience tested: A year and a half of ten thousand aftershocks. University of Canterbury. Accessed 24 November 2020

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

Verb Phrase Translation in English to Tamil Machine Translation

Vijay Sundar Ram, R and Sobha, Lalitha Devi

AU-KBC Research Centre, MIT campus of Anna University, Chennai
sundar@au-kbc.org

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

S.No	English Sentence	Tamil translation using	
		Google Translation	Bing Translation
1	He has been writing.	அவர் எழுதுகிறார்.	அவர் எழுதி வருகிறார்.
2	He may be writing.	அவர் எழுதுகிறார்.	அவர் எழுதி இருக்கலாம்.

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 'எழுதுகிறார்'. Whereas in Bing translation, 'progressive marker' 'கொண்டு' 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 phase 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 fie is as follows.

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

Translated Sentence:
<Sent Id=24 lang='ta'> வந்தியத்தேவன் எழுந்திருக்கவில்லை . </Sent>

VP Map Index:
<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 ‘எழுந்திருக்கவில்லை’. 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.

Table 2. Statistics of the training and testing data

S.No	Language Pair	Details	Training Data	Testing Data
1	English to Tamil	Number of Sentences	1992	1000
		VPs indexed	2267	1869

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*, Σ, Γ, *I*, *F*, δ) such that:

- *Q* is a finite set, the set of *states*;
- Σ is a finite set, called the *input alphabet*;
- Γ 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 ε 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+கொண்டு_வரு_கிற்+PNG

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

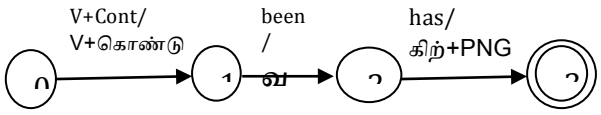


Figure 1: Sample FST

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

Table 3: Sample State table

Initial State	Next State	Input	Output
0	1	V+Cont	V+கொண்டு
1	2	been	வரு
2	3	has	கிற்+PNG
3			

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+கொண்டு', then it move to state 2, when input symbol 'been' is received and produced an output 'வரு' and finally when it receives the input symbol 'has', it moves to the end state 3, producing the output 'கிற்+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 4: Scoring Criteria

S.No	Criteria	Score
1	Completely Correct	4
2	TAM and PNG Correct	3
3	Correct root and TAM partially correct	2
4	Correct root and wrong TAM	1
5	Completely Incorrect	0

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

Table 5: Scores in each Criterion

Criteria Score				
4	3	2	1	0
462	568	219	216	404

Precision and recall scores are presented in the table 6

Table 6: Performance Measures

S.No	Precision	Recall
1	55.92%	78.38%

The results show that the number of correctly identified TAM is higher and it has led to better performance scores. The number of unattempt verb phrases are significant. This clear 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

1. Sobha .L., Pralayankar, P., Menaka, S., Bakiyavathi, T., Vijay Sundar Ram, R.., Kavitha, V.: Verb transfer in a Tamil to Hindi machine translation system. In: Asian Language Processing (IALP), 2010 International Conference on. pp. 261–264. IEEE (2010)
2. Vijay Sundar Ram and Sobha Lalitha Devi. (2018). “VPT-IL: Verb Phrase Translation in Machine Translation: English - Tamil and Hindi - Tamil @ FIRE 2018 - An Overview”, In the Forum for Information Retrieval and Evaluation-2018, DAIICT, Gandhinagar, India.

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

Named Entity Recognition in Tamil from Code Mix Social Media Text

Pattabhi, R K Rao and Sobha, Lalitha Devi

[1] AU-KBC Research Centre, MIT Campus of Anna University, Chennai, India
pattabhi@au-kbc.org

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 etal., 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 1. Corpus Statistics

Language	No. of Tweets	No. of NEs
Tamil-English	4576	2454

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:1234567890123456789
User_Id:1234567890

NE_TAG:ORGANIZATION
NE Raw String:SonyTV
Index:43
Length:6

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 filed 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.

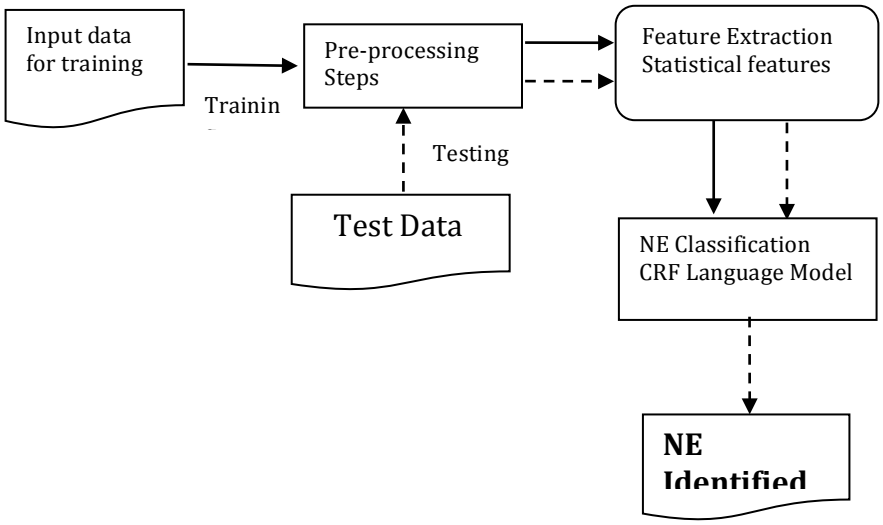


Fig 1. System Architecture – Process Flow Diagram

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

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

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, fj (Y_{i-}1, Yi, 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 fj(y_{i-}1,yi,x,i) = 1 if y_i="LOCATION" and the suffix of i_{th} word is “land”; otherwise 0; If the weight λ_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 2. System Results – Average Scores

Language	Precision	Recall	F-measure
Tamil-English	78.56 %	64.66%	70.93%

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

1. José Ramom Pichel Campos, Iñaki Alegría Loinaz, Nora Aranberri, Aitzol Ezeiza, Víctor Fresno. 2014 TweetLID@SEPLN 2014, Girona, Spain, September 16th, 2014. CEUR Workshop Proceedings 1228, CEUR-WS.org 2014
2. Mark Dredze, Tim Oates, and Christine Piatko. 2010. “We’re not in kansas anymore: detecting domainchanges in streams”. In *Proceedings of the 2010 Conferenceon Empirical Methods in Natural LanguageProcessing*, pages 585–595. Association for ComputationalLinguistics.
3. Hege Fromreide, Dirk Hovy, and Anders Søgaard.2014. “Crowdsourcing and annotating ner for twitter#drift”. *European language resources distributionagency*.
4. H.T. Ng, C.Y., Lim, S.K., Foo. 1999. “A Case Study on Inter-Annotator Agreement for Word Sense Disambiguation”. In *Proceedings of the {ACL} {SIGLEX} Workshop on Standardizing Lexical Resources {(SIGLEX99)}*. Maryland. pp. 9-13.
5. Preslav Nakov and Torsten Zesch and Daniel Cer and David Jurgens. 2015. Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015).
6. Nakov, Preslav and Rosenthal, Sara and Kozareva, Zornitsa and Stoyanov, Veselin and Ritter, Alan and Wilson, Theresa. 2013. SemEval-2013 Task 2: Sentiment Analysis in Twitter. *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*
7. Rajeev Sangal and M. G. Abbas Malik. 2011. Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (SANLP)
8. Aravind K. Joshi and M. G. Abbas Malik. 2010. *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing* (SANLP). (http://www.aclweb.org/anthology/W10-36)

9. Rajeev Sangal, Dipti Misra Sharma and Anil Kumar Singh. 2008. *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*. (http://www.aclweb.org/anthology/I/I08/I08-03)
10. Pattabhi RK Rao, CS Malarkodi, Vijay Sundar R and Sobha Lalitha Devi. 2014. Proceedings of Named-Entity Recognition Indian Languages track at FIRE 2014. http://au-kbc.org/nlp/NER-FIRE2014/
11. Wallach, H.M. (2004). Conditional random fields: An introduction Technical Reports (CIS), *MSCIS-04-21*
12. Lafferty, J., McCallum, A. & Pereira, F. (2001). Conditional Random Fields for segmenting and labeling sequence data. *ICML-01*. 1, 282-289.

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

Balance Scale to Disambiguate 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.

2. Balance Scale: A novel Approach

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.

S.No.	Telugu NN-NOM	Tamil NN-NOM	Gloss
1.	lēkuMḍā	illāmal/ inṟi	‘without’
2.	kākuMḍā	allāmal/anṟi	‘except’

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

3.1. Te. lēkuMḍā and Ta. illāmal/inṟi `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.

1.	Te.	nēnu	tāḷaMcevi	lēkuMḍā	talupu	tīs-	ā-	nu.	
		I.NOM	key.NOM	without	door.ACC	open-	PST-	1.SG.	
	Ta.	nāṉ	cāvi	[illāmal/inṟi]	katav-	ai.t	tiṟa-	nt-	ēṉ.
		I.NOM	key.NOM	without	door-	ACC	open-	PST-	1.SG.
	‘I opened the door without a key.’								

2.	Te.	nēnu	lēkuMḍā	vāḍu	sinimā-	ku	veḷḷ-	ā-	ḍu.
		I.NOM	without	he.NOM	cinema-	DAT	go-	PST-	3.SG.M.
	Ta.	nāṉ	[illāmal/inṟi]	avaṉ	paṭattu-	kku.p	pō-	ṉ-	āṉ.

		I.NOM	without	he.NOM	cinema-	DAT	go-	PST-	3.SG.M.
		`He Went to the cinema without me.'							

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,

3.	Ta.	vāḍu	iMṭ-lō	[lē-kuMḍā/	uMḍ-akuMḍā]	baiṭa	nivasiMc-ā-ḍu.	
		he.NOM	home-LOC	[be.NEG-CPM/	be-NEG.CPM]	outside	stay-PST-3.SG.M.	
	Ta.	avaṉ	vīṭṭ-il	[il.l-āmal/	iru.kk-āmal]	veḷiyē	taṅk-	iṉ-āṉ
		he.NOM	home-LOC	[be.NEG-CPM/	be-NEG.CPM]	outside	stay-	PST-3.SG.M.
		`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).

4.	Te.	nēnu	tāḷaMcevi	[lēkuMḍā/	*uMḍa-	kuMḍā]	talupu	tīs-	ā-	nu.	
		I	key	[without/	be-	NEG.CPM]	door.ACC	open-	PST-	1.SG.	
	Ta.	nāṉ	cāvi	[illāmal/	*iru.kk-	āmal]	katav-	ai.t	tiṟa-	nt-	ēṉ.
		I	key	[without/	be-	NEG.CPM]	door-	ACC	open-	PST-	1.SG.
		`I opened the door without a key.'									

3.2. Te. kākuMḍā and Ta. allāmal/aṉṟi `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	veḷḷ-	ā-	ru.
		I.NOM	except	everyone	cinema-	DAT	go-	PST-	3.PL.H.
	Ta.	nāṉ	allāmal	aṉaivarum	paṭattu-	kku	pō-	ṉ-	ārkaḷ.
		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').

6.	Te.	kōḍiguḍḍu	pilla	[kā-kuMḍā/	avv-akuMḍā]	tapp-a-du.
		egg.NOM	Young one	[become.NEG-CPM/	become-NEG.CPM]	escape from-NEG.FUT-3.SG.N.
		`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	[kākuMḍā/	tappa]	aMdarū	sinimā-	ku	veḷḷ-	ā- ru.
		I.NOM	except		everyone	cinema-	DAT	go-	PST-3.PL.H.
	Ta.	[nāṉ	allāmal/	eṉṉ-ai.t	tavira]	aṉaivarum	paṭattu-	kku	pō-ṉ-ārkaḷ.
		[I.NOM	except/	I-ACC	except]	everyone	cinema-	DAT	go-PST-3.PL.H.

		`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.

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

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

4.1. Te. guMḍā and Ta. vaḻiyāka `through (a place)'.

A noun[+place] occurs with the postposition guMḍā in Telugu and vaḻiyāka in Tamil with a verb[+motion] to express through the place.

8.	Te.	mūrti	aḍavula	guMḍā	prayāṇaM	cēs-	ā-	ḍu.	
		Murti	forests.OBL	through	travel	do-	PST-	3.SG.M	
	Ta.	mūrtti	kaṭukaḷ-	(iṉ)	vaḻiyāka	payaṇaM	cey-	t-	āṉ.
		Murti	forests.OBL-	(GEN)	through	travel	do-	PST-	3.SG.M

		`Murti traveled through forests.'

The postposition va̲liyāka in Tamil is a grammaticalized form of va̲li `way' + āka `ADV'. When the noun va̲li occurs as a complement of the postposition va̲liyāka, it is optionally dropped. In the case of Telugu, dāri `way' and guMḍā `through' can co-occur, since the postposition is not derived from the former one.

9.	Te.	kukka	nā	iMṭi	dāri	guMḍā	parigett-	iM-	di.
		dog.NOM	my	house.OBL	way	through	run-	PST-	3.SG.N.
	Ta.	nāy	e̲n̲	vīṭṭu	va̲liyāka	ōṭi-	y-	atu	
		dog.NOM	my	house.OBL	through	run-	PST-	3.SG.N.	
		`The dog ran through the way of my house.'							

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

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

Balance scale for noun[+place]-va̲liyāka 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.

10.	Te.	kāluvala-	guMḍā	polāla-	ku	nīḷḷu	kaḍu-	tunnā-	mu.
		canal.OBL	through	fields-	DAT	water.PL	irrigate-	PRS-	2.PL.
	Ta.	kālvāy-	va̲liyāka	nilaṅkaḷ-	ukku	taṇṇir	pāyccu-	ki̲r̲-	ōm.
		canal.OBL	through	fields-	DAT	water.SG	irrigate-	PRS-	2.PL

		`We irrigate fields through canal.'

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]-va̲liyāka 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 -ā̲rppōl is used in such constructions.

11.	Te.	nā-	ku	eduru- guMḍā	mūrti	vacc-	ā	-ḍu.
		I-	DAT	In front of	Murti	come-	PST-	3.SG.M.
	Ta.	e̲n-	akku	etir.tt- ā̲rppōl	mūrtti	va-	nt-	ā̲n.
		I-	DAT	In front of	Murti	come-	PST-	3.SG.M.
		`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 -ā̲rppōl 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]-ā̲rppōl in Tamil:
-āka > -il iruntu> nōkki >> -il/-ē << va̲liyāka < *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.

12.	Te.	nēnu	tālaMcevi-	dvārā	talupu	teric-	ā-	nu.	
		I.NOM	key	with	door-	open-	PST-	1.SG.	
	Ta.	nāṉ	cāvi	mūlam	katav-	ai.t	tiṟa-	nt-	ēṉ.
		I.NOM	key	with	door-	ACC	open-	PST-	1.SG.
		`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 guMḍa 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.

13	T	ṭīcaru	caduvu	paṭl	āme	āsakti	ni	gamani	ā-.	ḍu

.	e.			a		-		Mc-		
		teach er	study.O BL	?	she.O BL	intere st-	AC C	notice-	PS T-	3.SG. M.
	Ta.	āciriy ar	paṭippu	mēl	avaḷ	ārvatt-	ai	kavaṉi-	tt-	ār.
		teach er	study.O BL	on	she.O BL	intere st-	AC C	notice-	PS T-	3.SG. M.
		`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).

14.	Te.	mūrti-	ki	ḍabbu	paṭla	āśa	uMdi.	
		Murti-	DAT	money	towards	desire	have	
	Ta.	mūrti-	kku	paṇatt-	iṉ	mēl	ācai	uḷḷatu.
		Murti-	DAT	money-	GEN	on	desire	have
		`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).

15.	Te.	mūrti	nā-	paṭla	kōpaM-	[gā/	tō]	unn-	ā-	ḍu.
		Murti	I.OBL	towards	angry-	[ADV/	ASS]	be-	PST-	3.SG.M.
	Ta.	mūrtti	eṉ	mēl	kōpam-	[āka/	ōṭu]	iru-	kkiṟ-	āṉ.
		Murti	I.OBL	on	angry-	[ADV/	ASS]	be-	PST-	3.SG.M.
		`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'.

16.	Te.	mūrti	rāDa	vaMka	cūs-	ā-	ḍu.	
		Murti	Radha.OBL	towards	see-	PST-	3.SG.M.	
	Ta.	mūrtti	rātā(.v	-iṉ)	pakkam	pār-	tt-	āṉ.
		Murti	Radha.OBL-	(GEN)	towards	see-	PST-	3.SG.M.
		`Murti saw towards Radha.'						

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].

17.	Te.	mūrti	nā-	kūḍa	iMṭi-	ki	vacc-	ā-	ḍu.
		Murti.NOM	I.OBL-	ASS	house-	DAT	come-	PST-	3.SG.M.
	Ta.	mūrtti	eṉ-	kūṭa	vīṭṭ-	ukku	va-	nt-	āṉ.
		Murti.NOM	I.OBL-	ASS	house-	DAT	come-	PST-	3.SG.M.
		`Murti came along with me to the house.'							

6. Te. prakāraM and Ta. paṭi `according to'

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

18.	Te.	nānna	iṣṭa	prakāraM	peḷḷi	jarig-	iM-	di.
		father	wish.OBL	according to	marriage	happen-	PST-	3.SG.N.
	Ta.	appā.v-	iṉ	viruppa(tt-	iṉ)	paṭi	kalyāṇam	naṭa-nt- atu.

		father-GEN	wish.	(GEN)	according to	marriage.NOM	happen-PST-3.SG.N.	
		`According to the desire of my father, the marriage has taken place.'						

7. Te. veMbaḍi/ veMṭa and Ta. u\=taṉ/ ōṭu/ piṉṉāl/ -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].

19.	Te.	nā	[veMbaḍi/veMṭa]	kumār	vas-	tunnā-	ḍu.	
		I.OBL	along with	Kumar	come-	DUR-	3.SG.M.	
	Ta.	eṉ.ṉ-	uṭaṉ	kumār	varu-	kiṟ-	āṉ	
		I	ASS	Kumar	come-	DUR-	3.SG.M.	
		`Kumar comes along with me.'						

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 piṉṉāl occurs with nouns in Tamil.

20.	Te.	pōlīsu	doMga	[veMbaḍi/ veMṭa]	parigett-	ā-	ḍu.
		police	thief	behind	run-	PST-	3.SG.M.

	Ta.	kāvalar	tiruṭaṉ	piṉṉāl	ōṭi-	ṉ-	ār.	
		police	thief	behind	run-	PST-	3.SG.H.	
		`The police ran behind the thief.'						

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

*tōpāṭu > *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)).

21.	Te.	mūrti	vīDi	[veMbaḍi/veMṭa]	naḍic-	ā-	ḍu.
		Murti	street.OBL	along	walk-	PST-	3.SG.M.
	Ta.	mūrtti	teru.v-	ōṭu	naṭa-	nt-	āṉ.
		Murti	street-	ASS	walk-	PST-	3.SG.M.
		`Murti walked along the street.'					

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

Nouns[+bodyparts] 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.

22.	Te.	kaḷḷa	[veMbaḍi/veMṭa]	nīru	kāru-	toM-	di.
		eyes.OBL	from	water	pour-	DUR-	3.SG.N.
	Ta.	kaṇkaḷ-	il iruntu	kaṇṇīr	vaḻi-	kiṟ-	atu.
		eyes-	from	tear	pour-	PRS-	3.SG.N.
		`Tears pour out from the eyes.'					

Balance scale for nouns[+body parts]-veMbaḍi/ veMṭa in Telugu:
*tōpāṭu > *kūḍa >> veMbaḍi/veMṭa << nuM\bdi
`ASS' > `along' >> `along' << `from'

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.

23.	Te.	vāḍu	iMṭi	paṭṭuna	unn-	ā-	ḍu.
		he.NOM	home.OBL	at	be-	PST-	3.SG.M
	Te.	avaṉ	viṭṭ-	il	iru-	kkiṟ-	āṉ.
		he.NOM	home.OBL	LOC	be-	PST-	3.SG.M
		`He is at home.					

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.

24.	Te.	nuvvu	vēḷa	paṭṭuna	tin-	āli.	
		you.NOM	time.OBL	in	eat-	FORC.	
	Ta.	nī	nērattu-	kku	cāppiṭ-	a-	vēṇṭum.
		you.NOM	time-	DAT	eat-	INF-	FORC.
		`You have to eat on time.'					

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.

25.	Te.	nēnu	reMḍu	ēḷḷa	mēra(ku)	selavu	peṭṭ-	ā-	nu.
		I.NOM	two	years.OBL	upto	leave	keep-	PST-	1.SG.
	Ta.	nāṉ	iraṇṭu	varuṭaṅkaḷ-	ukku	viṭumuṟai	eṭu-	tt-	ēṉ.
		I.NOM	two	years-	DAT	leave	take-	PST-	1.SG.
		`I took leave for two years.'							

Postpositions mēra(ku) in Telugu and paṭi in Tamil express the function of `in accordance with'.

26.	Te.	mā	amma	kōrika	mēra(ku)	caduvukonn-	ā-	nu.	
		our	mother.OBL	wish.OBL	In accordance with	study-	PST-	1.SG.	
	Ta.	eṅkaḷ	ammā.v-	iṉ	viruppa.p	paṭi	paṭi-	tt-	ēṉ.
		our	mother-	GEN	wish.OBL		study-	PST-	1.SG.
		`I studied in [accordance/ compliance] with our mother's wish.'							

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 in no context discovered. In this way, postpositions from Telugu to Tamil is substituted to obtain better results in the
current MT.

References

Dorr, Bonnie Jean. 1993. *Machine Translation: a View from the Lexicon*. Massachusetts: MIT press.

Krishnamurti, Bh. 2003. *The Dravidian Languages*. Cambridge: Cambridge University Press.

Krishnamurti, Bh. & J. P. L. Gwynn. 1985. *A Grammar of Modern Telugu*. Delhi: Oxford University Press.

Lehmann, Thomas. 1993. *A Grammar of Modern Tamil*. Pondicherry: Pondicherry Institute of Linguistics and Culture.

Parameswari, K. & G. Uma Maheshwar Rao. 2012. 'Syntactic Divergences: Description and Solution in Machine Translation', *International Workshop on Syntax* at CALTS, University of Hyderabad, Hyderabad, February.

Parameswari, K., G. Uma Maheshwar Rao, Sreenivas N.V. & M. Christopher. 2012c. 'Development of Telugu-Tamil Bidirectional Machine Translation System: A Special Focus on Case Divergence'. In 11th *International Tamil Internet Conference.,* 180–191. Annamalainagar: Annamalai University.

Ramanarasimham, P. 2006. *An Intensive Course in Telugu*, vol. 9. Mysore: Central Institute of Indian Languages.

Ramarao, Chekuri. 1975. *Telugu vākyam ['The Telugu Sentence']*. Hyderabad: AP Sahitya Academy.

Shanmugam, S. V. 2013. *The Morphosyntax of the Dravidian Languages*. Mysore:Central Institute of Indian Languages.

Subbarao, K. V. 2012. *South Asian Languages: A Syntactic Typology.* Cambridge: Cambridge University Press.

Complexties in Developing Tamil-Brahmi Script OCR: An Analysis

M Monisha[1*], V S Felix Enigo[2]
[1]Research Scholar, [2]Associate Professor
[1,2]Department of Computer Science and Engineering
[1,2]SSN College of Engineering, Chennai
[1]moni.munivel@outlook.com, [2]felixvs@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 inscriptional 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 langauge. 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 langauges. 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 theTamil-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.

Sl.No	Region	Name of the Region	Sites	Inscriptions
1	Northern Region	Thondai Nadu	4	4
2	Southern Region	Pandiya Nadu	24	71
3	Western Region	Cera Nadu	5	22
4	Eastern Region	Chola Nadu	1	1

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.

Sl.No	Period		Sites
1	Early Period	3th BCE - 1st BCE	Aiyarmalai, Alagaramalai, Arittapatti, Karungalakudi, Kilavalavu, Kilkuyilkudi, Kongarpuliyangulam, Mangulam, Marukaltalai, Mettupatti,Mudalaikulam, Muttupatti,

			Pulimankombai, Sittannavasal, Thathappati, Tirumalai, Tirupaunkundram, Tiruvadavur, Varichiyur, Vikkramangalam
2	Middle Period	1st BCE – 2nd CE	Alagaramalai, Ammankoyilpatti, Edakal, Jambai, Mamandur, Mannarkoil, Muttupatti, Pugalur, Tiruchirapalli, Tirumalai, Tirupaunkundram, Tondur
3	Late Period	3rd CE – 4th CE	Anaimalai, Arachalur, Kudumiyanmalai, Kunnakudi, Nekanurpatti, Pugalur

Table 2. Chronological arrangement of Tamil-Brahmi inscriptions

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.

Sl. No	Site Name	District	Riverbank	Excavation period	No. of Potteries	Dated on palaeographic evidence	Languages
1	Arikamedu	Virampatnam	Ariyankuppam	1941-44 1947-50 1989-92	66	2BCE-3CE	Tamil, Prakrit and Pali
2	Uraiyur	Tiruchirapalli	cauvery	1965-69	20	1 BCE	Tamil
3	Kodumanal	Erode	Noyyal	1985-86 1989-90 1996-97	170	2 BCE	Tamil, Prakrit and Pali
4	Alagankulam	Rameswaram	vaigai	1986-1987	Graffiti marks	2BCE-1CE	Tamil, Pali

5	Salihundam	srikulam	Vamsadhara	1950	69	1 BCE	Prakrit
6	Pattanam	ernakulam	Periyar	2010-2011	3	4 CE	Tamil
7	Keezhadi	Sivaganga	Vaigai	2015 - Present	72 till 2019	6 BCE	Tamil

Table 3. Various sites of Tamil-Brahmi pottery inscription in Tamil Nadu

Fig 2. Potsherds from excavation

Some potsherds of Tamil-Brahmi inscriptions was also found in excavations carried out in countries like Sri lanka, Egypt, Oman and Thailand and the details are given in Table 4.

Sl.No	Site Name	Country	Potsherds dated	Language
1	Poonagari, Jaffna	Sri Lanka	2 BCE	Tamil
2	Quseir-al-Qadim	Egypt	1 CE	Prakrit
3	Berenike	Egypt	1 CE	Prakrit
4	Khor Rori	Oman	1 CE	Tamil
5	Krabi	Thailand	2 CE	Tamil

Table 4. Potsherds found in various countries

Few pottery inscriptions was found from Surface exploration in the sites of Attur, Coimbatore, Jambai, Kallupatti, Pappinayakkanpatti, Odaikalpalaiyam, Sivakasi and Marungur.

3. Characteristics of Tamil-Brahmi Script

The inscriptional Tamil laguage 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 complexties 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 complexites are listed in Table 5.

S.No	Difficulties	Examples
1	Similar Shapes by Writers	(pha), (va), (ca)
2	Unnecessary Strokes	(u), (ra), (ray)
3	Minor Variation	(nga), (u)
4	Shape Variation	(u), (na)
5	Angle Variation	(u), (pa)
6	Character Continuity	(daa), (ja)
7	Similar in Look	(ni), (ngi)
8	Curve Variation	(pa), (pha)
9	Double Characters	(Shri)

Table 5. Complexties 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 syllablary 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 langauge. 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.

Sl.No	Rule	Description	Inscription Text	Anticipated Text
1	Consonant in Tamil Brahmi	Consonants doubled	Koṭupitōṉ ceypita upu tacaṉ	koṭṭupi(t)tōṉ ceypi(t)ta u(p)pu ta(c)caṉ
2	Consonants in the loanwords	Consonants doubled in loanwords (words borrowed from Prakrit)	upacan dhamam puta atittanam	upa(c)can dha(m)mam pu(t)ta atitta(n)nam
3	Consonants as a grammatical	Consonants doubled as per the	Amanan katumi	ama(n)nan katu(m)mi

	feature	orthographic conventions	kurumakal	kuru(m)makal
4	Doubling of the final consonant	The final consonant of a monosyllabic stem is doubled when followed by a vowel	av an	av(v)an
5	Doubling of the initial consonant	The initial consonant of the succeeding part is doubled with a consonant	Ce-kanti Tavan-ur-pin-an	Ce-(k)kanti Tavan-ur-(p)pin-an

Table 6. Doubling of Consonants

4.2 Issues identified by Analysing the writings inTamil-Brahmi inscriptions

By analysing the Tamil-Brahmi inscription it was found that in places of single word, it occurs as double word segments. Such word segments should be merged to a single word to be recognized by OCR. The rules to join such word segments is given in Table 7.

Sl.No	Analytical Writing	Types	Inscription text	Anticipted Text
1	A consonant followed by a vowel	Between word segments	Kutal-ur	Kutalur
		Between word and suffix	Katal-an	Katalan
		Between word segment / suffix	Ay-am Per-ay-am	ayam perayam
2	Vowel followed by a vowel	Between word-segments	Karu-ur	karu(v)ur
		Between word and suffix	Ati-on	ati(y)on

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.

Sl.No	Changes	Description	Rule	Inscription Text →

				Anticipated Text
1	Loss of voicing	Caused by a voiceless contiguous consonants	g → k d→ t b → p	**g**anaka → **(k)**anaka **d**anam → **(t)**ana kutum**b**ika → kutum**(p)**ikan
2	Loss of aspiration	Lost at the beginning of a stressed syllable when the voiceless plosives	gh→k jh→c th→tt	**gh**atika → **(k)**atika upa**jh**a → upa**(c)**an adi**th**ana → ati**(tt)**anam
3	Loss of anusvara (m)	Characterizes sound produced by lowering the soft palate	End with M	**Da**na**m** → **Ta**na
4	Loss of h	Deletion of voiceless glottal fricative or /h/ sound	a→ha	**a**riti → **ha**riti **a**ritan →**ha**ritan

Table 8. Orthographic changes in loanwords

4.4 Issues in Tamil-Brahmi Morphophonemics

Morphophonomics 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, iḷam + kāyipaṉ read as iḷam kāyipaṉ not as iḷaṁkāyipaṉ. Table 9 shows the effect of assimilation in Tamil-Brahmi script.

Sl.No	Assimilation	Rules	Inscription Text	Anticipated Text
1	Regressive assimilation	m+k→ ṁk	iḷam + kāyipaṉ perum + kaṭum + kōṉ	iḷaṁkāyipaṉ peruṁkaṭuṁkōṉ
		m+c → ñc	iḷam+caṭikaṉ neṭum+caḻiyaṉ	iḷañcaṭikaṉ neṭuñcaḻiyaṉ
		ḷ+k → ṇk	veḷ + kasipan veḷ + kācipaṉ	veṇkasipan veṇkācipaṉ
		l+p → np	veḷ + palli	Veṇpalli
		l+n → nn	eḷ + nai(ney)	Ennai
2	Mutual assimilation	l + tt → rr	vempiḷ + tt + ur	vempirr-ur

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.

Sl.No	Addition	Rule	Inscription Text	Anticipated Text
1	Addition of glide	Glide –y	karui+a elai+ur Paḷḷi + ai	Karuiya elaiyur paḷḷiyai
		Glide –v	illa + on kuna +in iru + ar	illavon kunavin iruvar
2	Addition in sandhi	–m → -n	pa+m+kat+a	panka(t)ta

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.

Sl.No	Elision Rule	Inscription Text	Anticipated Text
1	stem final vowel before a succeeding vowel	kaṇa + atikaṉ toṇṭi + ilavōṉ	kaṇatikaṉ toṇṭilavōṉ
2	stem final suffix (a)m followed by the noun	Kalam + nel pāṇitam + vāṇikaṉ	kalanel pāṇitavāṇikaṉ
3	stem final suffix (a)n followed by the noun	araṭṭan + kāyipaṉ kaṇakaṉ + ataṉ	araṭṭkāyipaṉ kaṇakataṉ
4	stem final suffix (a)n before a genitive suffix	Cēntaṉ + ā Kuviraṉ + ā	Cēntaṉā Kuviraā

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.

Sl.No	Rule	Inscription Text	Anticipated Text
1	Disappearance of Ikaram	makaṉ + ḷaṅkaṭuṅkō kaṭuṅkōṉ +ḷaṅkaṭuṅkō	makaṉ iḷaṅkaṭuṅkō kaṭuṅkōṉ iḷaṅkaṭuṅkō
2	Disappearance of Aṉ suffix	kaṇaka ataṉ kuvira antai	kaṇakkaṉ ataṉ kuviraṉ antai
3	Sprinkling of ikara yakara (Ikara yakara iṟuti viravutal)	paḷi'iy karu'iya vaḻuttiy	paḷḷi karukki vaḻutti iya
4	Writing suffix separately (விகுதியைத் தனியாகப் பிரித்தெழுதுதல்)	koṭiy-avaṉ veḷ-aṟai koṭi'-ōr cēvit-ōṉ tiṭi'-il	Koṭṭiyavaṉ veḷḷaṟai koṭṭiyōr ceyvittōṉ tiṭiyil

5	Soul_Body letters (உடல் மேல் உயிர் வந்து ஒன்றுவது இயல்பு)	par'-acu pākaṉ-ūr pēr'-ay-am	paracu pākaṉūr pērayam

Table 12. Other Grammar issues

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 recognization 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

1. Abhishek Tomar, Minu Choudhary, Amit Yerpude, “Ancient Indian Scripts Image Pre-Processing and Dimensionality Reduction for Feature Extraction and Classification: A Survey”, International Journal of Computer Trends and Technology (IJCTT) – Volume 21 Number 2 – Mar 2015.
2. U. Bhattacharya, S. K. Ghosh, S. K. Parui, “A Two Stage Recognition Scheme for Handwritten Tamil Characters”, Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 1, pp. 511-515. IEEE, 2007.
3. Giridharan. R, Vellingiriraj.E.K, Balasubramanie.P, “Identification of Tamil Ancient Characters and Information Retrieval from Temple Epigraphy Using Image Zoning”, International Conference on Recent Trends in Information Technology (ICRTIT), pp. 1-7. IEEE, 2016.
4. R.Indra Gandhi, K.Iyakutti, “An Attempt to Recognize Handwritten Tamil Character Using Kohonen SOM”, International Journal of Advance d Networking and Applications, Volume: 01 Issue: 03 Pages: 188-192 (2009).
5. Mahadevan, Iravatham. 2014. *Early Tamil Epigraphy: From the Earliest Times to the Sixth Century C.E*, Revised and Enlarged Second Edition,

Vol I, Tamil Brahmi Inscriptions, Central Institute of Classical Tamil, Chennai.
6. Murugaiyan, Appasamy. "Emergence of Tamil as Epigraphic Language: Issues in Tamil Historical Linguistics." (2019).
7. K. Punitharaja, and P. Elango, “Tamil Handwritten Character Recognition: Progress and Challenges”, International Journal of Control Theory and Applications 9, no. 3 (2016): 143-151.
8. Sonal P.Patil, Priyanka P. Kulkarni, “Online Handwritten Sanskrit Character Recognition Using Support Vector Classification”, International Journal of Engineering Research and Applications, Vol. 4, Issue 5(Version 1), May 2014, pp.82-91.
9. Sugata Das, Sekhar Mandal, Amit Kumar Das, “Binarization of stone inscripted documents”, In: 2015 IEEE international conference on computer graphics, vision and information security (CGVIS).

தரவக வழித் தமிழ்ச் செவ்வியல் நூல்கள் ஆய்வு : சிக்கல்களும் தீர்வுகளும்

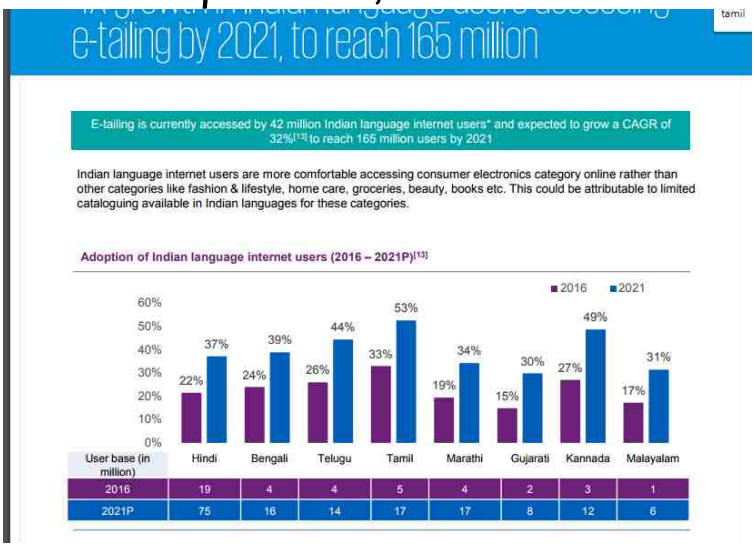
முனைவர் இரா.அகிலன்
நிரலாளர்
செம்மொழித் தமிழாய்வு மத்திய நிறுவனம்
சென்னை

1. முன்னுரை

ஒழுங்குமுறையுடன் அதிக அளவில் தேர்வு செய்யப்பட்டுக் கணினியில் சேமிக்கப்பட்ட இயற்கையான நடைகளை உடைய உரைகள், பல்வேறு பனுவல்களின் தொகுப்புத் தரவகம் எனப்படும். இக்கட்டுரை தகவல் இயற்கை மொழி ஆய்வுக் கருவிகளான சங்க இலக்கியச் சொல்லடைவி, தொடரடைவு, உருபனியல் பகுப்பான் மற்றும் உருவாக்கி, சங்க இலக்கியத் தேடுபோறி போன்ற சங்க இலக்கியக் கருவிகள் உருவாக்கத்தின் அடிப்படையான சங்க இலக்கியத் தரவகம் உருவாக்கம் பற்றியும் அத்தரவகத்தின் வழி செவ்வியல் நூல்களின் ஆய்வுகள் பற்றியும், அதை உருவாக்கும் போது ஏற்படும் சிக்கல்கள் மற்றும் தீர்வுகள் பற்றியும் விவாதிக்கின்றது.

2. மொழி இன்றைய நிலை

மொழி காலந்தோறும் பல்வேறு தடங்களில் பயணிக்கிறது. கல்வெட்டுகள், செப்பேடுகள், ஓலைச்சுவடிகள், தாட்சுவடிகள், நூல்கள் என்ற வரிசையில் வெவ்வேறு ஊடகங்களில் பயணித்து வளர்ந்து வந்துள்ளதைக் காணலாம். இப்பொழுது மொழி கணினியிலும் கைபேசியிலும் பயணிக்கிறது. கூகுள் நிறுவனம் 2017ஆம் ஆண்டு உலகமொழிகளுக்கான ஒரு வரைவு அறிக்கையை தாயாரித்தது. அந்த அறிக்கையில் வரும் நூற்றாண்டுகளில் மொழிகளின் பயன்பாடு, ஆதிக்கம், செயலாக்கம் போன்றவை இணைத்தின் பயன்பாட்டுத் தரவுகளின் அடிப்படையிலேயே அமையும் என்றும் மேலும் வரும் 2021ஆம் ஆண்டில் அந்த மொழிகளின் இணையவழிப் பயன்பாடு எவ்வாறு அமையும் என்பதையும் குறிப்பிட்டு அந்த அறிக்கை (படம் 1) வெளியிடப்பட்டது.



படம் 1. கூகுள் நிறுவன அறிக்கை

இந்த அறிக்கையின் படி நாம் மேற்க்கொள்ள வேண்டியது ஒருமொழியை பரப்ப அந்த மொழியை தகவல் தொழில்நுட்பத்தின் வழி பயன்பாட்டில் கொண்டுவருவது மட்டுமே ஆகும். ஆகவே இணையத்தில் தமிழ் மொழியை எழுதுவது, தொழில் நுட்பக்கருவிகளை தமிழ் மொழியில் பயன்படுத்துவது போன்ற காரணிகள் நம் உயர்தனிச் செம்மொழியை அடுத்த நிலைக்கு எடுத்துச் செல்லும் உந்து கருவிகள் என்பதை நிறுவுகிறது.

3. செவ்வியல் தமிழ் தரவகம்

தமிழ்ச் செவ்வியல் நூல்கள் 41 அவையாவன இலக்கண நூல்கள் - 2 (தொல்காப்பியம், இறையனார் களவியல்), சங்க இலக்கியம் - 10 (பத்துப்பாட்டு, எட்டுத்தொகை), பதினெண் கீழ்க்கணக்கு - 18 காப்பியங்கள் - 2, சிற்றிலக்கியம் – 2 ஆகும். இந்த 41 செவ்வியல் நூல்களில் இடம்பெற்றுள்ள பாடல்களின் எண்ணிக்கை 7,849, அடிகளின் எண்ணிக்கை 57666, மூலபாடத்தில் ஒருமுறை பயின்று வரும் சொற்களின் எண்ணிக்கை 1,59,025, மூலபாடத்தில் பயின்றுவரும் மொத்த சொற்கள் எண்ணிக்கை 228098 ஆகும். இத்தரவகத்தைப் பயன்படுத்தி சங்க இலக்கியங்களுக்கான சொல்லடைவி, தொடரடைவு, உருபனியல் பகுப்பான் மற்றும் உருவாக்கி, சங்க இலக்கியத் தேடுபோறி போன்ற சங்க இலக்கியக் கருவிகள் உருவாக்கலாம்.

4. தரவக வழி செவ்வியல் நூல்கள் ஆய்வு

தரவக வழி மொழித் தொழில்நுட்பக் கருவிகள் உருவாக்கத்திற்கு ஆய்வுகளுக்குப் பயன்படுவதற்குரிய மென்பொருள் உருவாக்கம் இரண்டு வகைகளில் நடைபெறுகிறது

1. தரவுகள் வழி உருவாக்கம்

2. விதிகள் வழி கருவிகள் உருவாக்கம்

தரவுகள் வழி தொழில்நுட்பக் கருவிகள் உருவாக்கமானது இணையதளங்களிலும், நூல்வடிவிலும் உள்ள தரவுகளைப் பயன்படுத்தித் தொழில்நுட்பக் கருவிகளை உருவாக்கலாம். விதிகள் வழி தொழில்நுட்பக் கருவிகள் உருவாக்கமானது மொழியியல் நுட்பத்தைப் பயன்படுத்திக் கணினி புரிந்துகொள்ளும் வகையிலான மொழியியல் விதிகளை உருவாக்கி அதன் மூலம் தொழில்நுட்பக் கருவிகளை உருவாக்கலாம். இதற்கு அதிகம் தரவுகள் தேவைப்படுவது இல்லை. ஆயினும் மேற்கூறிய இரண்டு வழிகளிலும் தொழில்நுட்பக்கருவிகள் உருவாக்கம் செய்ய, விரிதரவு மற்றும் குறியீட்டு விரிதரவு ஆகிய இரண்டும் அடிப்படை தேவையாய் இருக்கிறது.

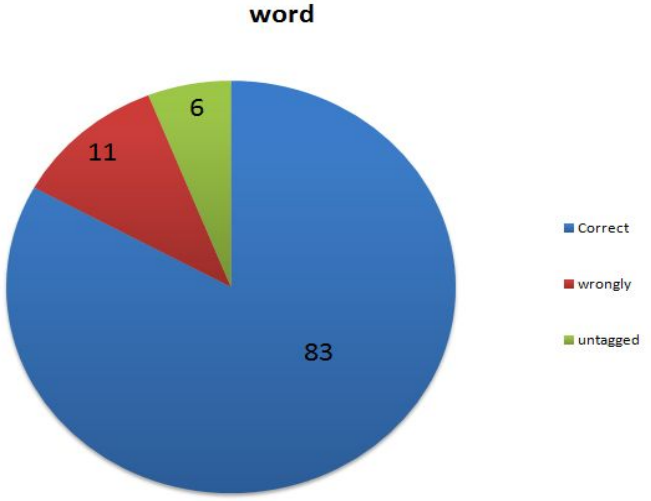
பொதுவாக மொழியியல் விதிகளைப் பயன்படுத்திச் சங்க இலக்கியத்திற்கான உருபனியல் பகுப்பான் உருவாக்கும் பொழுதும் இதற்கு ஒரு அகராதி தேவைப்படுகிறது. அந்த அகராதியும் இலக்கணக் குறியீட்டு முறையில் அமைந்து இருப்பது அவசியமாகிறது. உருபனியல் பகுப்பான் உருவாக்கத்திற்கான பன்மைக் குறியீட்டிற்கான மொழியியல் விதிகளைப் பயன்படுத்தி உருவாக்கும் முறை:

1. வேர்ச்சொல் அகராதி சோதனை { 'ஆம்' குறியீடு அடையாளப் படுத்துகை }:முடிவு
2. இல்லை {ஒட்டுக்களைச் சோதி}
3. ஒட்டு 'கள்' {ஒட்டுக்களைப் பிரித்தல் குறியீடு அடையாளப் படுத்துகை }
4. வேர்ச்சொல் அகராதி சோதனை
5. 'ஆம்' குறியீடு அடையாளப் படுத்துகை : முடிவு
6. ஒட்டு 'ங்கள்' {ஒட்டுக்களைப் பிரித்தல் குறியீடு அடையாளப் படுத்துகை }
7. முதல் எழுத்து 'ங்' நீக்கம்
8. {குறியீடு அடையாளப் படுத்துகை }
9. சொல்லில் இறுதியில் சேர்க்க (ம்)
10. குறியீடு அடையாளப் படுத்துகை
11. ஒட்டு 'ற்கள்' {ஒட்டுக்களைப் பிரித்தல் குறியீடு அடையாளப் படுத்துகை }
12. முதல் எழுத்து 'ற்' நீக்கம்
13. குறியீடு அடையாளப் படுத்துகை
14. சொல்லில் இறுதியில் சேர்க்க (ல்)
15. குறியீடு அடையாளப் படுத்துகை
16. ஒட்டு 'ட்கள்' {ஒட்டுக்களைப் பிரித்தல் குறியீடு அடையாளப் படுத்துகை }
17. முதல் எழுத்து 'ட்' நீக்கம்
18. குறியீடு அடையாளப் படுத்துகை
19. சொல்லில் இறுதியில் சேர்க்க (ல்)
20. குறியீடு அடையாளப் படுத்துகை
21. முடிவு

அட்டவணை 1. செவ்வியல் நூல்களுக்கான தரவகம்

சொல்	குறியீட்டுத் தரவகம்
காந்தளும்	காந்தள்\NC உம்\CLI
பேதையின்	பேதை\NC இன்\MOBL
மன்ற	மன்ற\CCP man̲ra\CCP
மென்	மென்\ADJ men̲\ADJ
காண்மார்	காண்\VB + மார்\VF
பதின்	பதின்\ CRD
யாக்கைக்கு	யாக்கை\NC க்கு\MDAT
ஈன்றாள் īn̲r̲āḷ	ஈன்\VB ற்\MPT ஆள்\PNG
மொழிப molipa	மொழிப\MFT
ஆஞ்சுவானை āñcuvān̲ai	அஞ்சு\VB வ்\MFT ஆன்\PNG ஐ\MACC

இவ்வாறு தரவகத்தைப் பயன்படுத்தி விதிகளை உருவாக்கி சங்க இலக்கிய நூல்களை ஆராயும் போது 41 இலக்கியளுக்கான உருபனியல் பகுப்பான் 83 சதவிகிதம் சரியான முடிவுகளை அளிக்கிறது. 11 சதவிகிதம் தவறான முடிவுகளையும், 6 சதவிகதம் சொற்களை இயந்திரத்தால் அடையாளம் கண்டுகொள்ள முடியவில்லை



படம் 3. சங்க இலக்கிய நூலகளுக்கான தரவக வழி கருவி முடிவுகள்

5. சிக்கல்கள்

தரவகத்தைப் பயன்படுத்தி விதிகளை உருவாக்கி அதன் வழியாக சங்க இலக்கியத்திற்கான மென்பொருள்களை உருவாக்கும் போது பல்வேறு சிக்கல்கள் எழுகின்றன. அகம் என்ற சொல் வெவ்வேறு சூழல்களில்

வித்தியாசமாக செயல்படுகிறது. சில சந்தர்ப்பங்களில், சொல் மற்றும் தொடரியல் அமைப்பு ஒரே மாதிரியாக இருக்கும். இலக்கண வகை என்பது வார்த்தையின் தொடரியல் கட்டமைப்பைப் பொறுத்தது. இந்த வகையான சொற்கள் மிகவும் சிக்கலானவை. அத்தகைய தொடர்கள் மற்றும் சொற்களை இயந்திரத்தால் பகுப்பாய்வு செய்ய முடியாது.

தான் **அகம்** புகாஅன் பெயர்தல் இன்மையின் (தொல்-பொருள். 1057-3)

மனை **அகம்** புகாஅக் காலை யான (தொல்-பொருள். 1081-3)

இதில் நூற்பா 1057-3-ல் 'அகம்' என்பது 'மனம்' என்னும் பொருளில் பெயர்ச்சொல்லாக வருகிறது. நூற்பா 1081-3-ல் 'அகம்' என்பது பின்னுருபாக வருகிறது.

அ நாள் கொண்டு இறக்கும் இவள் அரும் பெறல் உயிரே (கலித்.5:19)

கண்டி நுண் கோல் கொண்டு களம் வாழ்த்தும் (பதிற்றுப்.43:27)

வெண் கோடு கொண்டு வியல் அறை வைப்பவும் (நற்.114)

இதில் பாடல் 5:19-ல் 'கொண்டு' என்பது 'உள்ளே'. என்னும் பொருளில் பின்னுருபாக வருகிறது. பாடல் 43:27-ல் 'கொண்டு' என்பது 'உள்ளே'. என்னும் பொருளில் பின்னுருபாக வருகிறது பாடல் 114-ல் 'கொண்டு' என்பது 'எடு'. என்னும் பொருளில் 'வினை' யாக வருகிறது இது போன்ற சொற்களை இயந்திரத்தால் பிரிக்க முடியாது.

செவ்வியல் நூல்களுக்கான கருவிகள் உருவாக்கும் பொழுது ஒரு தொடரிலுள்ள சொற்கள் மட்டுமின்றி அத்தொடரின் பொருளும் மிகவும் இன்றியமையாதது. தொடரின் பொருளை முழுமையாக அறிந்தால் மட்டுமே சொற்களைத் தெளிவான முறையில் பிரிக்க முடியும். சங்க இலக்கியத்தில் எது சொல்; எது சொல் ஆகாது என்பதை வரையறுத்துக் கூறுவது சிக்கலானது. சில அறிஞர்கள் தனிச்சொல்லைக் கூட்டுச் சொல்லாகவும், கூட்டுச் சொல்லைத் தனிச்சொல்லாகவும் பிரித்துள்ளனர். கணிப்பொறி மூலம் தொடரடைவு / சொல்லடைவு உருவாக்கும் தெளிவான சொற்பிரிப்பு வரைமுறையுடன் சொற்களைப் பிரிக்க வேண்டும்.

6. முடிவுரை

ஒவ்வொரு காலகட்டத்திற்கான தொழில்நுட்பங்களையும் உள்வாங்கிக்கொண்டு பயன்பாட்டில் அதில் இடம்பெறும் தன்மை கொண்டது தமிழ்மொழி. சங்க இலக்கிய விழுமியங்களைத் தாங்கிய செம்மொழியாகவும், நீதிக்காலகட்டத்தில் அறமொழியாகவும், பக்திகாலகட்டத்தில் சமயமொழியாகவும், அறிவியல் காலகட்டத்தில் அறிவியல் மொழியாகவும், கணினிக் காலத்தில் கணினித்தமிழாகவும் தன்னைப் புதுப்பித்துக்கொண்டு இனிவரும் எதிர்காலத் சமூகத் தொழில்நுட்ப வளர்ச்சிக் காலகட்டத்திலும் செயலாற்றும் தன்மைக்

கொண்ட மொழியாக விளங்குகிறது. மொழித் தொழில்நுட்பத்தின் வழி தமிழ் மொழி சிறப்பை அடைவதற்கு மிகுதியான உழைப்பும் தமிழ் அறிஞர்கள், மொழியியல் அறிஞர்கள் மற்றும் கணினி வல்லுனர்களின் கூட்டு முயற்சி தேவைபடுகிறது. தொழில்நுட்பங்களை அவரவர் மொழியிலேயே பயன்படுத்த விழிப்பணர்வை ஏற்படுத்த வேண்டும். இவ்வாறு தமிழ் மொழியை தகவல் தொழில்நுட்பக்கருவிகளில் பயன்படுத்தி வரும் பொழுது மட்டுமே நம் உயர்தனிச்செந்தமிழ் மொழியை உலகெங்கும் பரப்பபடும்.

துணை நூல்கள் :

1. கோ. பழனிராஜன், லெ. ராஜேஷ், மு. முகமது யூனுஸ், இரா. அகிலன், “கைபேசிக் கலைச்சொல் அகராதி” இராசகுணா பதிப்பகம், டிசம்பர் 2016.
2. Prof. E.R.Naganathan, R.Akilan "Morphological Analyzer for Classical Tamil text - a Rule based approach " 2012, 12th International Internet conference, Annamalai University, Chidamparam.
3. கோ.பழனிராஜன், துணைப் பேராசிரியர், மொழியியல் துறை, கேரளா மத்தியப் பல்கலைக் கழகம், கேரளா “மொழித் தொழில்நுட்பம் ஓர் அறிமுகம் ” 2013 ‘கணினியியல் தொழில் நுட்பங்களும் சங்க இலக்கிய ஆய்வுகளும்’ தேசியக் கருத்தரங்கு, எஸ் ஆர் எம் பல்கலைக்கழகம், சென்னை
4. Natural Language Understanding by Allen J. – The Benjamins Publishing Company – 1995
5. Prof. E.R.Naganathan, R.Akilan "Morphological Analyzer for Classical Tamil text - a Rule based approach " Ph.D., Thesis, Bharathiar University, Coimbatore-2018

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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 Wn be a word of length n; Pi is a prefix of length i. Let C be a corpus of words. CPi is defined as the subset of C containing those words whose first i letters match Pi exactly. The **successor variety (Succ_Variety)** of Pi, 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, … , 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: ஆர்அம்ப்இத்த்இர்உக்க்இற்ஆன் ('start' - verb + Aux + present + 3rd-person singular masculine - 'he has started')

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

Starting Sequence	Ending Sequence	Succ_Variety	Pred_Variety	Segment_SV	Segment_PV	Stem	Suffix
ஆ	ர்அம்ப்இத்த்இர்உக்க்இற்ஆன்	10	2	1	0	ஆ	ர்அம்ப்இத்த்இர்உக்க்இற்ஆன்
ஆர்	அம்ப்இத்த்இர்உக்க்இற்ஆன்	1	2	0	0		
ஆர்அ	ம்ப்இத்த்இர்உக்க்இற்ஆன்	1	2	0	0		
ஆர்அம்	ப்இத்த்இர்உக்க்இற்ஆன்	1	2	0	0		
ஆர்அம்ப்	இத்த்இர்உக்க்இற்ஆன்	1	2	0	0		
ஆர்அம்ப்இ	த்த்இர்உக்க்இற்ஆன்	5	4	1	1	ஆர்அம்ப்இ	த்த்இர்உக்க்இற்ஆன்
ஆர்அம்ப்இத்	த்இர்உக்க்இற்ஆன்	1	3	0	0		
ஆர்அம்ப்இத்த்	இர்உக்க்இற்ஆன்	7	5	1	1	ஆர்அம்ப்இத்த்	இர்உக்க்இற்ஆன்
ஆர்அம்ப்இத்த்இ	ர்உக்க்இற்ஆன்	2	3	0	0		
ஆர்அம்ப்இத்த்இர்	உக்க்இற்ஆன்	1	4	0	0		
ஆர்அம்ப்இத்த்இர்உ	க்க்இற்ஆன்	4	6	1	0	ஆர்அம்ப்இத்த்இர்உ	
ஆர்அம்ப்இத்த்இர்உக்	க்இற்ஆன்	1	8	0	1		க்இற்ஆன்
ஆர்அம்ப்இத்த்இர்உக்க்	இற்ஆன்	3	2	1	0	ஆர்அம்ப்இத்த்இர்உக்க்	
ஆர்அம்ப்இத்த்இர்உக்க்இ	ற்ஆன்	2	5	0	0		
ஆர்அம்ப்இத்த்இர்உக்க்இற்	ஆன்	3	12	1	1	ஆர்அம்ப்இத்த்இர்உக்க்இற்	ஆன்
ஆர்அம்ப்இத்த்இர்உக்க்இற்ஆ	ன்	2	9	0	0		
ஆர்அம்ப்இத்த்இர்உக்க்இற்ஆன்		1	1	0	0	ஆர்அம்ப்இத்த்இர்உக்க்இற்ஆன்	

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 "ஆர்அம்ப்இ" has a peak successor varieties count, and marks the segment boundary. The starting sequences ஆர்அம்ப்இ, ஆர்அம்ப்இத்த், ஆர்அம்ப்இத்த்இர்உ, ஆர்அம்ப்இத்த்இர்உக்க் and ஆர்அம்ப்இத்த்இர்உக்க்இற் 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 **ஆன், க்இற்ஆன், இர்உக்க்இற்ஆன் and த்த்இர்உக்க்இற்ஆன்** 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.

ஆர்அம்ப்இ followed by {க்க்இற்ஆர், த்த்ஓம், ப்ப்ஆன், ய்உங்க்அள், Space} with count 5

ஆர்அம்ப்இத்த் followed by {அ, ஆ, இ, ஈ, உ, ஏ, ஓ} with count 7

ஆர்அம்ப்இத்த்இர்உ followed by {க், ந், ப், space} with count 4.

ஆர்அம்ப்இத்த்இர்உக்க் followed by {அ, இ, உ} with count 3

ஆர்அம்ப்இத்த்இர்உக்க்இற் followed by {அ, ஆ, ஓ} with count 3 in the selected corpus

The Highest successor count occurs at the beginning of a word. The letter 'ஆ' 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.

ஆர்அம்ப்இ + த்த்இர்உக்க்இற்ஆன்
ஆர்அம்ப்இத்த் + இர்உக்க்இற்ஆன்
ஆர்அம்ப்இத்த்இர்உ + க்க்இற்ஆன்
ஆர்அம்ப்இத்த்இர்உக்க் + இற்ஆன்
ஆர்அம்ப்இத்த்இர்உக்க்இற் + ஆன்

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

ஆர்அம்ப்இ + த்த் + இர்உ + க்க் + இற் + ஆன்

From the above segments, it is observed that the first segment produces stem of a given word form.

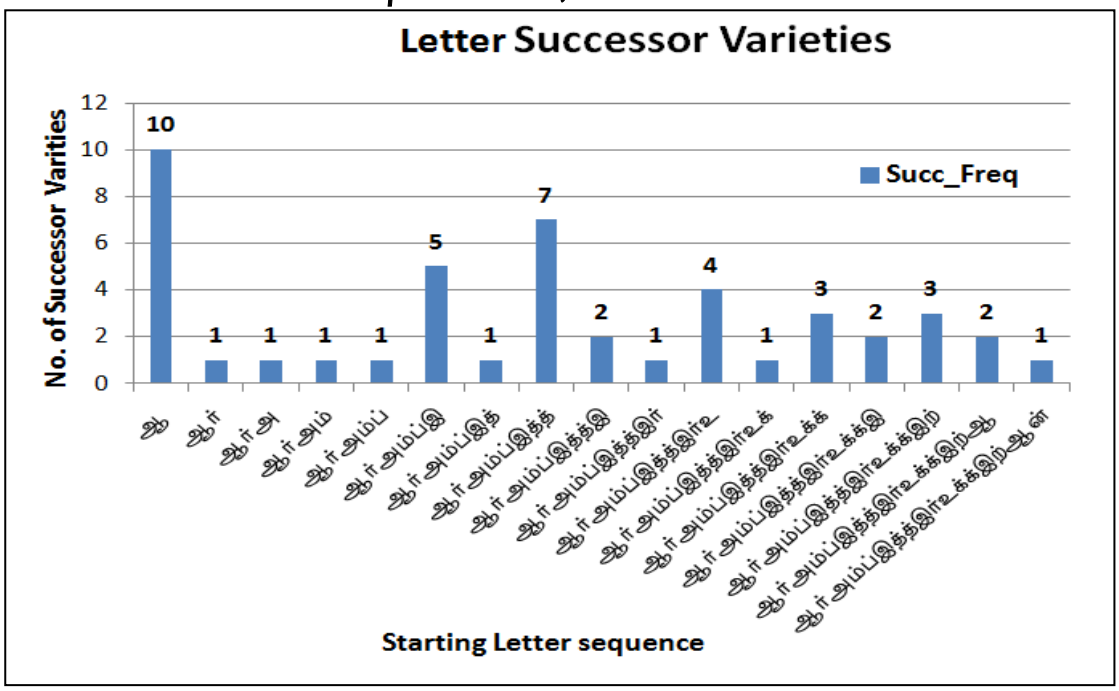


Figure 1 Letter Successor Variety counts

The **Letter** P**redecessor 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.

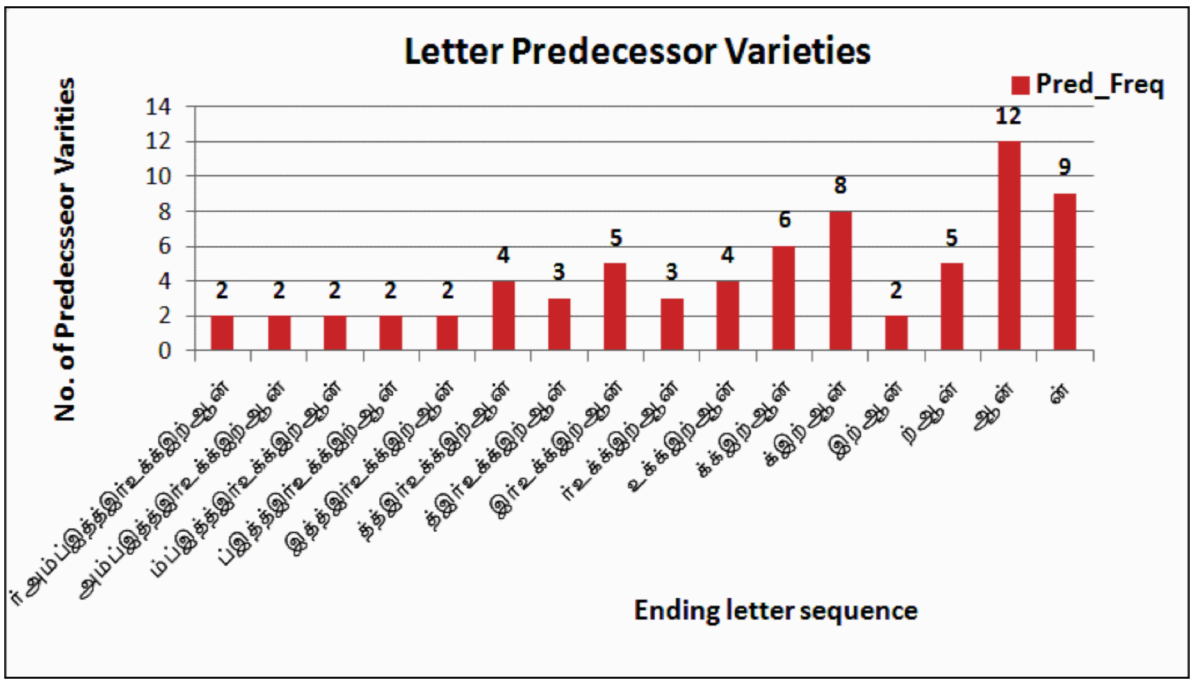


Figure 2 Letter Predecessor Variety counts

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 4,00,000 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 Segement_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.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

$$F\ Measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

Table 2 Sample morpheme segmentations

No.	Over all Word Segmentation	TP	FP	FN
	ஆர்அம்ப்இ+த்த்+இர்உ+க்க்இற்+ஆன் (Reference or Gold Segmentation)			
1	ஆர்அம்ப்இ+த்த்+இர்உ+க்க்+இற்+ஆன் (SV)	4	1	0
2	ஆர்அம்ப்இ+த்த்+இர்உக்+க்இற்+ஆன் (PV)	3	1	1
3	ஆர்அம்ப்இ+த்த்+இர்உக்க்இற்+ஆன் (SV AND PV)	3	0	1
4	ஆர்அம்ப்இ+த்த்+இர்உ+க்+க்+இற்+ஆன் (SV OR PV)	4	2	0
5	ஆர்அம்ப்இ+த்த்+இர்உ+க்க்இற்+ஆன் (Rule based)	4	0	0

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

Methods	Precision %	Recall %	F-Score %
Succ_Variety_Peak (SV)	80.74	68.34	74.00
Pred_Variety_Peak (PV)	72.23	74.18	73.19
Peak of Both SV AND PV	**86.99**	65.78	74.92
Peak of SV OR PV	66.02	**76.57**	70.91
RULE BASED	**95.63**	48.94	64.75

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.

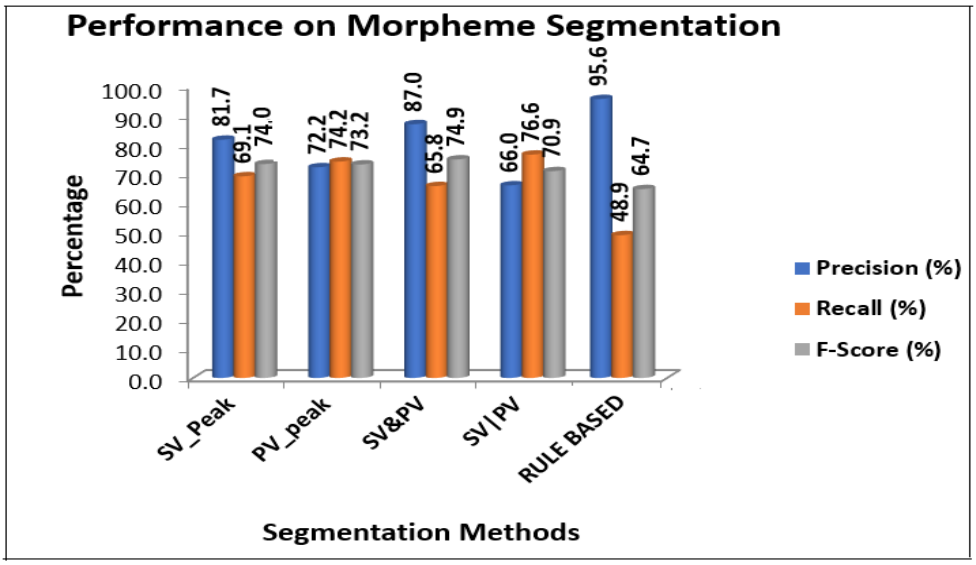


Figure 3. Performance Comparison of unsupervised methods (SV-PV) and rule-based method on morpheme segmentation

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

[1]. Z. Harris, "Structural Linguistics", University of Chicago Press, 2003.

[2]. M.A. Hafer and S.F.Weiss, "Word Segmentation by Letter Successor Varieties". Information Storage and Retrieval, vol. 10, pp. 371–385, 1974.
[3]. J. Goldsmith, "Unsupervised learning of morphology of a natural language", Computer Linguistics, vol. 27, pp. 153-198, 2001.
[4]. Mathias Creutz, "Unsupervised segmentation of words using prior distributions of morph length and frequency", In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Sapporo, Japan, pp. 280–287, 2003
[5]. M.Ganesan, "Morph and POS Tagger for Tamil" (Software), Annamalai University, Annamalai Nagar, 2009.
[6]. K. Rajan, V. Ramalingam, M. Ganesan, "Unsupervised Approach To Tamil Morpheme Segmentation", ***In Proceedings of Tamil Internet 2009, Cologne, Germany***, pp. 228-231, October 2009.
[7]. K. Rajan, V. Ramalingam, M. Ganesan, "Computational Approaches for Learning Inflections in Tamil", ***In Proceedings of Tamil Internet 2010***, ***Coimbatore***, pp. 183-189, June 2010.
[8]. K.Rajan, Machine Learning Techniques for Tamil Morphology (Unpublished Doctoral dissertation), Annamalai University, Annamalainagar, India, 2016.
[9]. M.Creutz and K.Lagus, "Unsupervised models for morpheme segmentation and morphology learning", ACM Trans. Speech Lang. Process, vol. 4, no. 1, 2007.
[10]. Taesun Moon, Katrin Erk, and Jason Baldridge, "Unsupervised morphological segmentation and clustering with document boundaries", Proc. Empirical methods in NLP, Singapore, pp. 668-677, 2009.

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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 கணனன (correct form is only 'கண்ணன்'), 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.

['கணனன', 'கணனன்', 'கணன்ன', 'கணன்ன்', 'கண்ணன', 'கண்ணன்', 'கண்ன்ன', 'கண்ன்ன்', 'க்ணனன', 'க்ணனன்', 'க்ணன்ன', 'க்ணன்ன்', 'க்ண்ணன', 'க்ண்ணன்', 'க்ண்ன்ன', 'க்ண்ன்ன்']

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

Algorithm:

Input:Tamil word **Wr** with potentially one or more missing pulli(s)

Output: Best alternative word **Wc**

1. Let **C** be list of all the consonants in order of occurrence in **Wr**
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, .. $2^{|C|}$-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, $\mathbf{P}(w_i|w_{i-1})$, picked off a standard reference - e.g. Tamil VU Dictionar or Project Madurai [2a,2b,4].
    a. $P(\mathbf{W}n) = \Pi_{i=2}^{n}\mathbf{P}(w_i|w_{i-1})$
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:

Incorrect	யாயும ஞாயும யாராகியரோ எநதையும நுநதையும எமமுறைக கேளிர
Correct	யாயும் ஞாயும் யாராகியரோ எந்தையும் நுந்தையும் எம்முறைக் கேளிர்

Misprinted Word	Alternates	Bigram Scores	Correct Word
யாயும	'யாயும்' 'யாயும'	20 16	யாயும்
ஞாயும	'ஞாயும்' 'ஞாயும'	20 16	ஞாயும்
யாராகியரோ	'யாராகிய்ரோ' 'யாராகியரோ'	36 31.94	யாராகியரோ
எநதையும	'எந்தையும்' 'எநதையும்' 'எந்தையும' 'எநதையும'	32 28.83 28 24.83	எந்தையும்
நுநதையும	'நுந்தையும்' 'நுநதையும்' 'நுந்தையும' 'நுநதையும'	36 32.839 23.0 28.839	நுந்தையும்
எமமுறைக	'எம்முறைக்' 'எமமுறைக்' 'எம்முறைக' 'எமமுறைக'	32.18 28.54 28.18 24.54	எம்முறைக்
கேளிர	'கேளிர்' 'கேளிர'	20 16	கேளிர்

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

1. W. H. Arden, “A Progressive Grammar of Common Tamil,” Soc. for Promoting Christian Knowledge (1910).
2. (a) Tamil Virtual Academy Dictionary http://www.tamilvu.org/library/dicIndex.htm (accessed Jun, 2020).
   (b) Solthiruthi module of Open-Tamil v0.96 https://pypi.org/project/Open-Tamil/ (accessed Jun, 2020).
3. Wikipedia entry on Sangam Poet Chembulapeyalneerar, https://ta.wikipedia.org/s/468 (accessed Jun, 2020).
4. M. Annamalai, T. Shrinivasan, “Algorithms for certain classes of Tamil Spelling correction,” INFITT Tamil Internet Conference, Chennai (2019).
5. JD Allen, et-al, “The Unicode Standard 5.0,” Addison-Wesley Professional (2012).

Appendix - A

```
# இந்த நிரல் துண்டு MIT உரிமத்தில் வெளியிடப்பட்டது
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)
```

COMPUTATIONAL ANALYSIS OF NOMINAL COMPOUNDS IN TAMIL

Rajendran Sankaravelayuthan
Amrita Vishwa Vidyapeetham, Coimbatore
rajushush@gmail.com

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:

[MAN]⟵(AGNT)⟵[BITE]⟶(OBJ)⟶[DOG]

[CAT]⟵(AGNT)⟵[SIT]⟶(LOC)⟶[MAT]

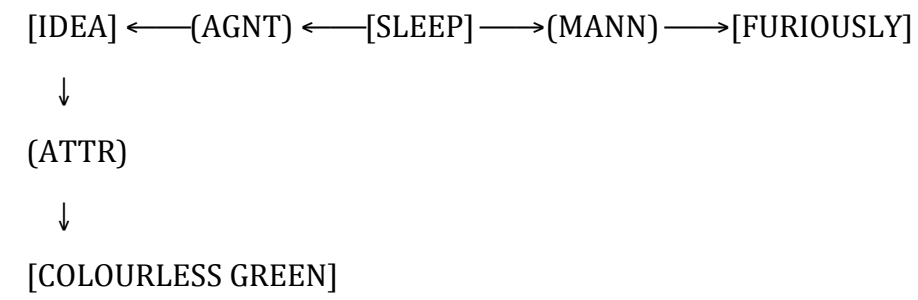
The information regarding tense and modalities are omitted in the above graphs. The square brackets enclose concepts and the braces enclose conceptual relations. "(AGNT)", "(OBJ)" and "(LOC)" denote respectively the conceptual relations (similar to case relations) AGENT, OBJECT and LOCATION. Though the 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:

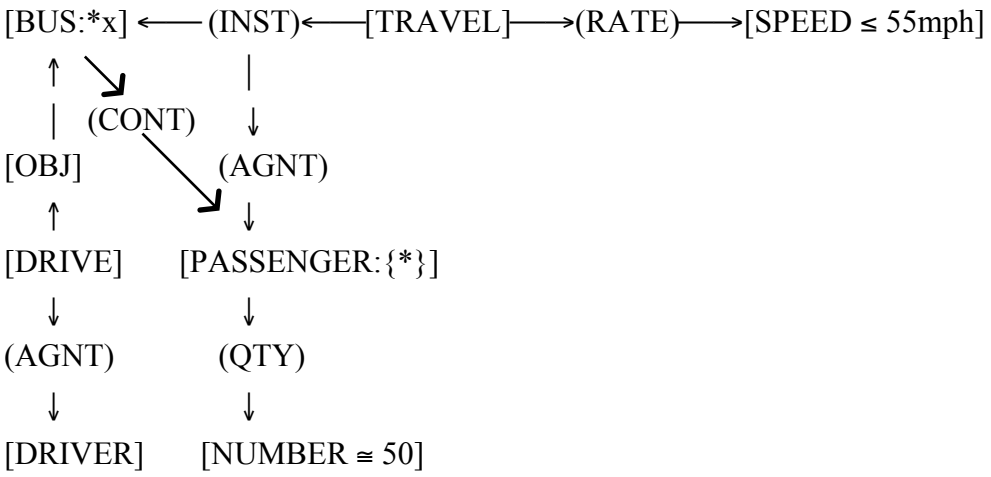
1. Arbitrary conceptual graphs which impose no constraints on permissible combinations.
2. Canonical graphs which enforce selectional constraints. They correspond to the case frames in linguistics and the category restrictions in philosophy.
3. Schemata which incorporate domain-specific knowledge about 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:



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):



(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:

[GERMS] ⟵(AGNT) ⟵[CAUSE] ⟶(OBJ) ⟶[DISEASE]

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

[DISEASE CAUSING GERMS]⟵(AGNT)⟵[CAUSE]⟶(OBJ)⟶[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:

[NEST] ⟵ (OBJ) ⟵ [MAKE] ⟶ (AGNT) ⟶ [CROW ⊂ 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:

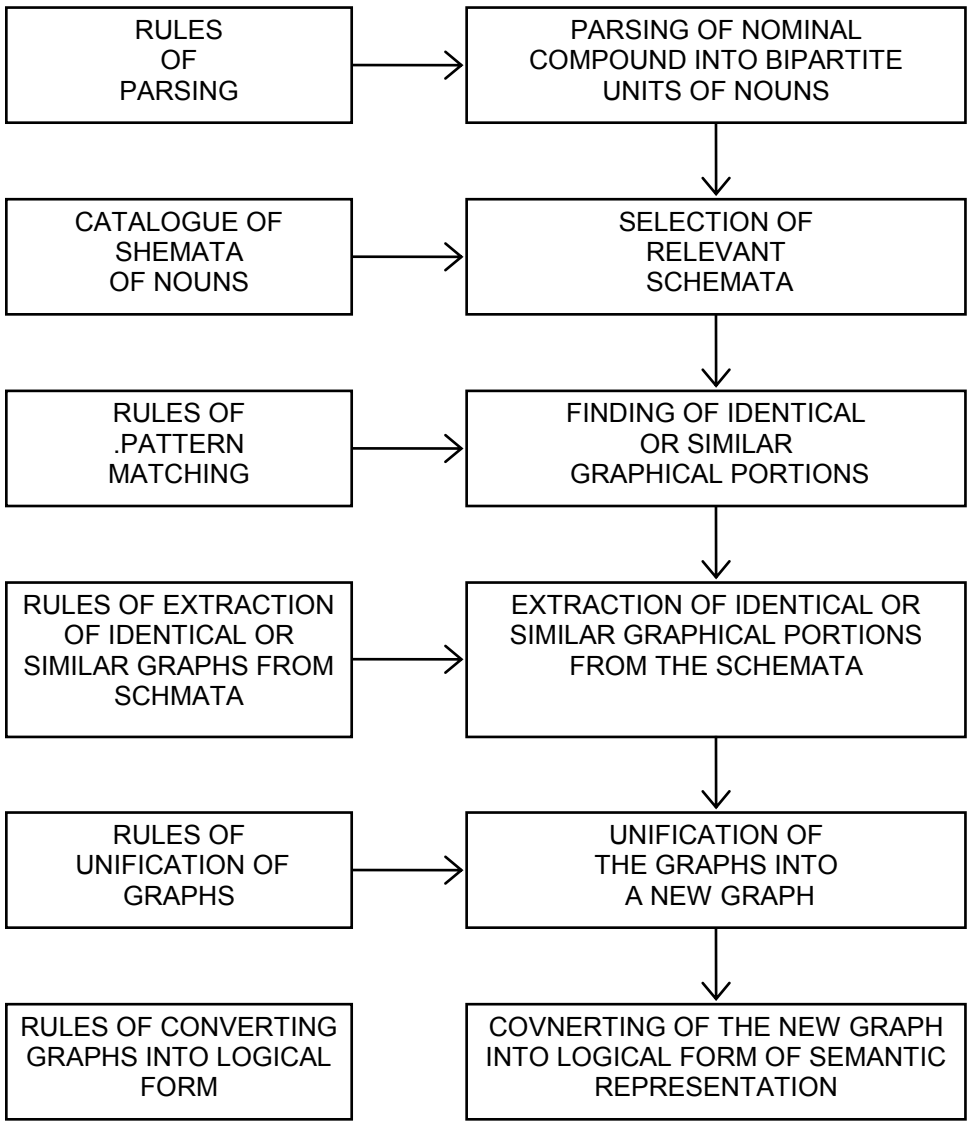
[TOY]⟵(OBJ)⟵[MAKE] ⟶(AGNT) ⟶[MAKER]
↓
(MADE UP OF)
↓
[CLAY ⊂ MATERIAL]

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 *kuruvi* = SPARROW and also the interpretation of *paRavaikkuuTu* '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:



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

இராசேந்திரன், ச. 2004. தமிழில் சொல்லாக்கம் [Word Formation in Tamil]. தமிழ்ப் பல்கலைக்கழகம், தஞ்சாவூர், 2004.

Allen, M. 1978. Morphological investigation. PhD dissertation, University of Connecticut.

Aronoff, M. 1976. Word formation in generative grammar. Cambridge, MA: MIT Press.
Bauer, L. 1983. English word-formation. Cambridge: Cambridge University press.
Botha, R.P.1968. The function of lexicon in transformational generative grammar.
Fabb, N. 1984. Syntactic affixation. PhD. dissertation, MIT.
Beard, R. 1988. On the separation of derivation from morphology: Towards a lexeme/morpheme-based morphology. Quaderni di Semantica 9, 3-59.
Borer, H. 1988. On the parallelism between compounds and constructs. Year book of morphology 1, 45-66.
Botha, R.P. 1968. The Function of lexicon in transformational generative grammar. The Hague: Mouton.
Botha, R.P. 1983. Morphological mechanism. Oxford: Pergamon press.
Chomsky, N. 1970. Remarks of nominalization. In: Jacobs, R and Rosenbaum, P. (eds.). Readings in English transformational grammar. Waltham, MA: Blaisdell.
Di Sciullo, A.M. and Williams, E. 1987. On the definition of word. Cambridge, MA: MIT press.
Fillmore, C.F. 1968. Case for case. In: Batch, E. and Harms , R. (eds.). 1968. Universal in Linguistic theory. New York: Holt, Rinehart and Wilson.
Lees, R.B. 1960. The Grammar of English nominalizations. International Journal of American Linguistics 26.
Lees, R.B. 1963. The Grammar of English nominalization. The Hague: Mouton.
Lees, R.B. 1970. Problems in the grammatical analysis of English nominal compounds. In: Bierwisch, M. and Heidolph, K.L. (eds.). 1970. Progress in Linguistics. The Hague: Mouton.
Levi, J.N. 1978. The Syntax and semantics of complex nominals. New York: Academic press.
Lieber, R. 1983. Argument linking and compounding in English. Linguistic Inquiry, 14, 251-86.
Rajendran, S. 1997. Grammatical Formalism and Computational Analysis of Nominal Compounds in Tamil. South Asian Language Review, vol. 7, no. 1, 1997, 27-46.
Rajendran, S. 2004. Strategies in the Formation of Compound nouns in Tamil, Language in India www.languageinindia.com 4:7, July, 2004.
Rajendran, S. 2005. A Comprehensive Study of Word formation in Tami. Language in India, October, 5:10, 2005, www.languageinindia.com.
Roeper, T. 1987. Implicit arguments and head-complement relation. Linguistic Inquiry 18, 267-310.
Roeper, T. 1988. Compound syntax and head movement. Year book of morphology. 1, 187-228.

Roeper, T. and Siegel, D. 1978. A Lexical transformation for verbal compounds. Linguistic Inquiry 9, 199-260.

Selkirk, E. 1982. The Syntax of words. Cambridge, MA: MIT press.

Sowa, J.F. 1984. Conceptual structures: Information processing in mind and machine. Reading: Addison-Wesley publishing company.

Spencer, A. 1991. Morphological theory: An Introduction to word structure in generative grammar. Oxford: Basil Blackwell.

Sproat, R. 1985. On deriving the lexicon. PhD dissertation. MIT.

Vijayavenugopal, G. 1979. Nominal composition in Tamil. Madurai: Madurai Kamaraj University.

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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 availble 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

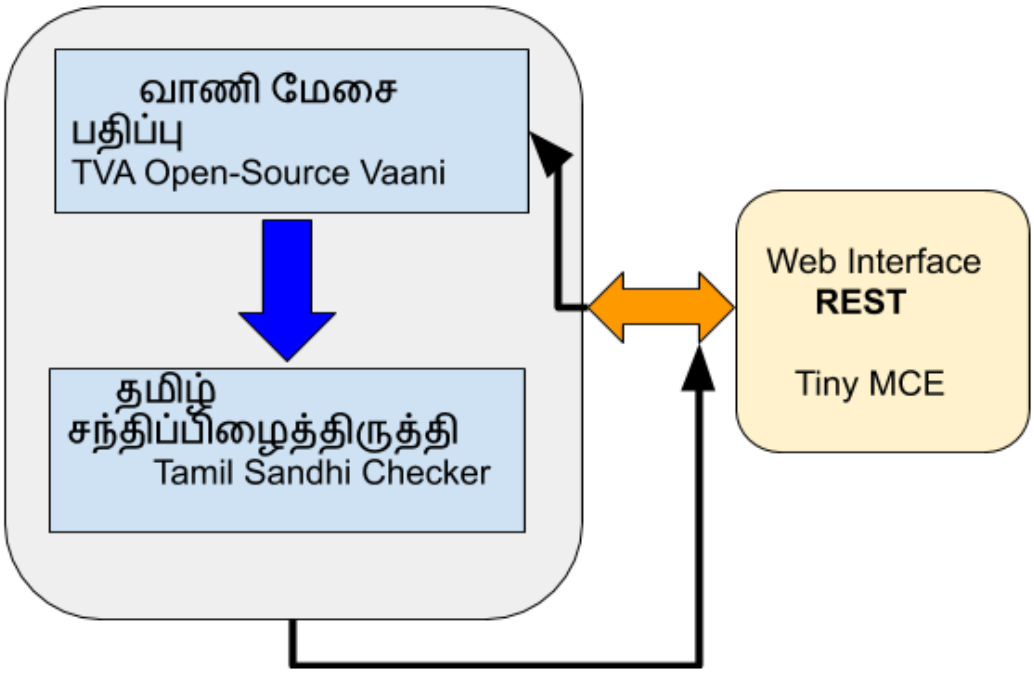


Fig. 1.a Organization of Python module tamilinayavaani; the REST interface portion is outside the module but a static method serves as callback handler for the web server.

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:

```
$ 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,

Usage Type	Code Sample
In-Memory Usage	from tamilinaiyavaani import SpellChecker flag,suggs=SpellChecker.REST_interface('வாழை','பழம்') expected=['வாழைப்', 'பழம்'] assert not flag assert expected[0]==suggs[0]
File based Usage	from tamilinaiyavaani import SpellChecker, SpellCheckerResult result = SpellChecker(fname).run() #fname is a full filename # result is a list of SpellCheckerResult objects.

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.

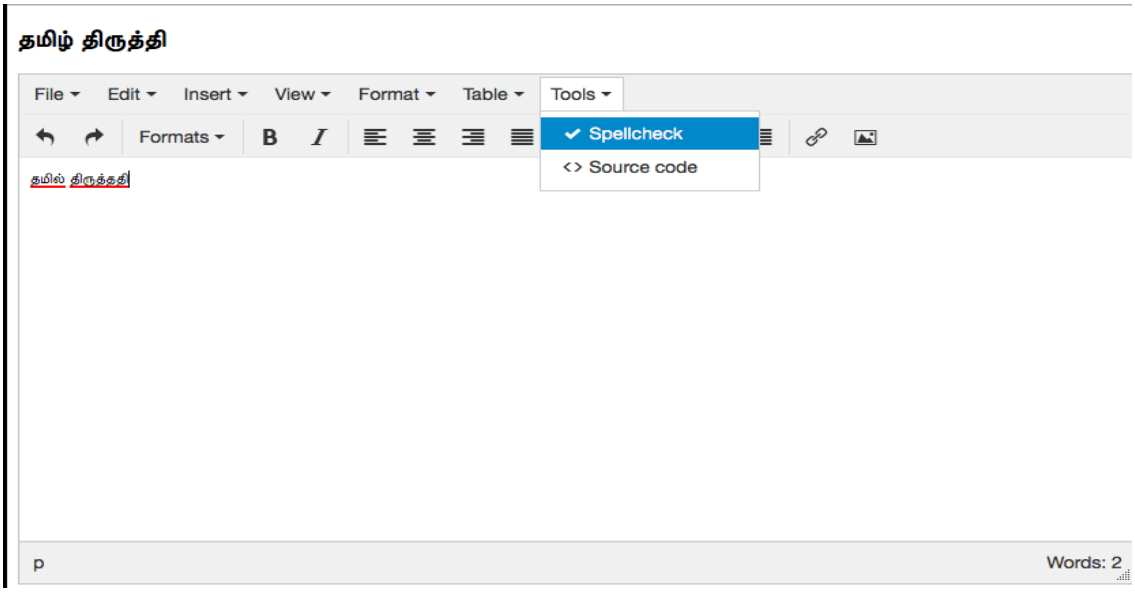


Fig. 2.a. Invoking the spell checker running in the Flask application server in backend.

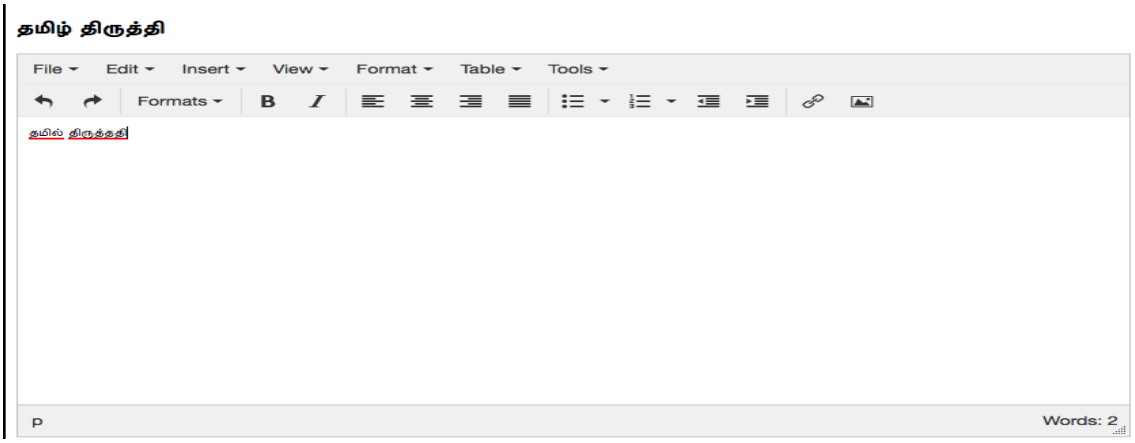


Fig. 2.b. Erroneous words indicated in the text/web-interface.

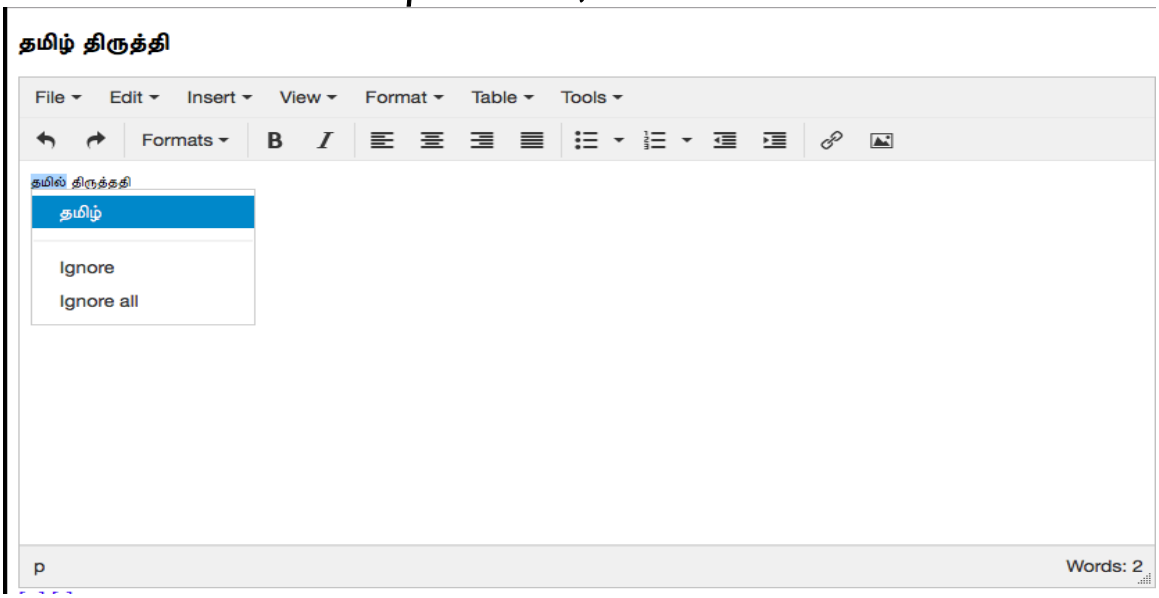


Fig. 1.c. Alternatives for first erroneous word

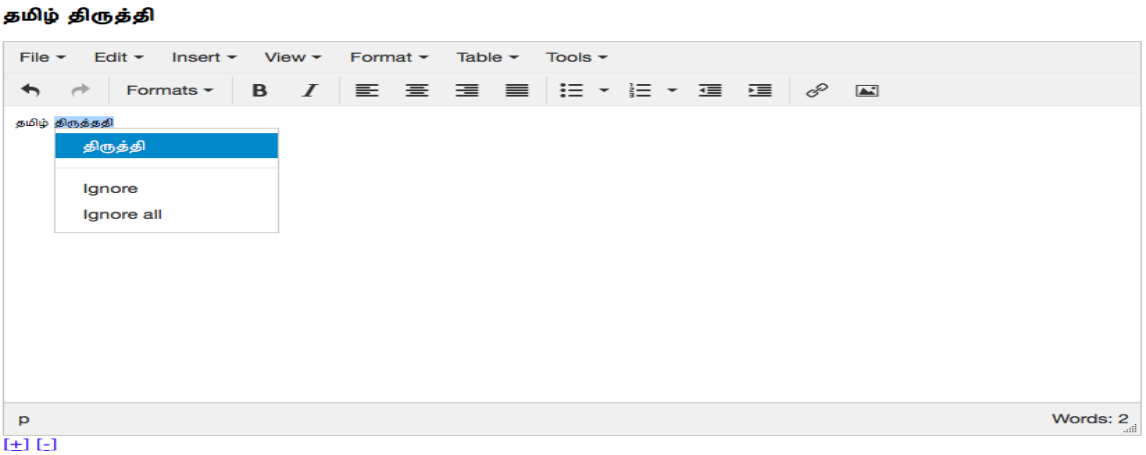


Fig. 2.d. Alternatives for second word after correcting first word.

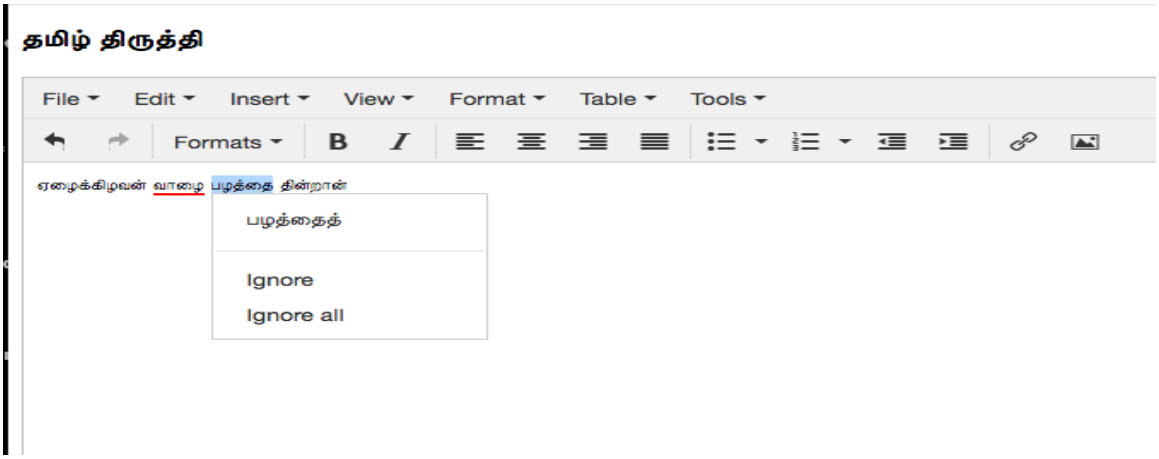


Fig. 3. Sandhi checker in action through the same web-interface.

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
2. Vaani - Tamil Spelling Checker by Neechalkaran: http://vaani.neechalkaran.com/ (accessed 2020).
3. நித்யா துரைசாமி, "தமிழுக்கான கட்டற்ற சந்திப் பிழை திருத்தி - உருவாக்கம், பயன்பாடுகள்," தமிழ் இணைய மாநாடு (2018).
4. PySpellchecker package https://pyspellchecker.readthedocs.io/en/latest/, (accessed 2020).
5. Python package for TamilInaya Vaani - https://pypi.org/project/tamilinayavaani/,(accessed 2020).
6. Hunspell, http://hunspell.github.io/ (accessed 2020)
7. Aspell, http://aspell.net/ (accessed 2020)
8. Langtool, https://languagetool.org/ (accessed 2020)
9. M. Annamalai, T. Shrinivasan, **Algorithms for certain classes of Tamil Spelling correction** (https://arxiv.org/abs/1909.10063) (2019)
10. Malaikannan S, et-al, "Improving spellchecker speed with BloomFilter," to appear in INFITT TIC (2020).
11. All Tamil Words - https://github.com/KaniyamFoundation/all_tamil_words
12. All Tamil Nouns - https://github.com/KaniyamFoundation/all_tamil_nouns
13. Series of blog posts on building an open source spellchecker for tamil - https://goinggnu.wordpress.com/category/spellchecker/

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

கலவைமுறைக் கற்றலில் தொழில்நுட்ப வளங்களின் பன்முகப் பயன்பாடு

முனைவர் இராமன் விமலன்
விரிவுரையாளர்
ஆசியமொழிகள் மற்றும் பண்பாடுகள் துறை
தேசியக் கல்விக் கழகம், நன்யாங் தொழில்நுட்பப் பல்கலைக் கழகம்
சிங்கப்பூர்.

முன்னுரை

உலகமயச் சூழலில் மானுட வாழ்வின் நோக்கும் போக்கும் மாற்றங்களுக்குட்பட்டு வருகின்றது. அறிவியல் தொழில்நுட்பத்தின் வளர்ச்சி வேகம் நாளுக்குநாள் மேம்பட்டு, புதுமைப்போக்குடன் வாழ்க்கை நடைபயில்கிறது. தமிழ்மொழியும் தொழில்நுட்பத்தின் செல்வாக்கிற்கேற்பத் தன்னைத் தகவமைத்துக்கொண்டு வாழ்ந்தும் வளர்ந்தும் வருகின்றது. கல்வித்துறையிலும் தொழில்நுட்பத்தை முன்னிலைப்படுத்தும் புதுப்புதுக் கற்பித்தல் முறைகள் தோன்றிக் கற்பித்தலைப் புதுமையாக்கியும் மகிழ்வூட்டும் கற்றலுக்கு வழிவகுத்தும் பயன்தரத்தக்க சூழலை ஏற்படுத்தியுள்ளன. அண்மையில் உலகையே புரட்டிப்போட்ட 'கொவிட்-19' தொற்றுநோய்ப் பரவல் கல்வித்துறைக்கும் பெரும் சவாவாக அமைந்தது. இந்நிலையில் 'இல்லம் சார்ந்த கற்றல்முறை' (Home based Learning) அறிமுகமானது. அதனைத் தொடர்ந்து, 'கலவைமுறைக் கற்றல்' (Blended Learning) அணுகுமுறை கற்றல்-கற்பித்தல் தளங்களுள் புகுந்து வெற்றிநடை போடுகின்றது. இத்தகு பயன்தரத்தக்க அடைவுநிலை தகவல் தொழில்நுட்ப வளங்களை அடிப்படையாகக் கொண்டமைந்தது என்பது குறிப்பிடத்தக்கது.

'கொவிட்-19' சூழலும் கலவைமுறைக் கற்றலும்

இந்நூற்றாண்டில் உலகையே நிலைகுலைய வைத்த ஒரு பெருநிகழ்வு 'கொவிட்-19' தொற்றுநோய்ப் பரவலாகும். இதனால், வளர்ந்துவரும் இளந்தலைமுறைக்கு அடித்தளமமைக்கும் கல்வித்துறை பெருமளவில் பாதிப்புக்குள்ளானது. பள்ளிகள் மூடப்பட்டுக் கற்றல்-கற்பித்தல் நடைமுறைகள் தடைபட்டன. மாணவர்களுக்கும் ஆசிரியர்களுக்குமான தொடர்புகள் அருகி, வீட்டில் முடங்கிக் கிடக்கும் அவலநிலை நீடித்தது. தொற்றுநோய்ப் பரவலை அழிக்கும் மருத்துவத் தீர்வுகள் கண்டறியப்படுவதில் ஏற்பட்ட தாமதப்போக்கும், அதிகரிக்கும் நோய்ப்பரவலும் கற்றல்-கற்பித்தலைப் புதுவகை இலக்கில் செலுத்த முனைந்தது. அதன் விளைவாக 'இல்லம்சார் கற்றல் முறை' (HBL) பயன்பாடு கண்டது. இந்நிலையில் தொழில்நுட்ப வளங்களைப் பயன்படுத்திக் கற்பிக்கும் செயல்முறைகள் சூடுபிடிக்கத் தொடங்கி, கற்றல்-கற்பித்தலில் தடையற்ற போக்கு நிலவியது. "கொவிட்-19 பரவலைத் தடுக்கும் வகையில் பள்ளிகள் மூடப்பட்டபோது இல்லம் சார்ந்த கற்றல் திட்டம் கற்றல் தடயின்றி நிகழ ஒரு முக்கியத் தளமாக மாறியது. பல

பெற்றோர்கள் தங்கள் குழந்தைகளின் கற்றல் முறைகள் மற்றும் கல்விப் பழக்கவழக்கங்களைப் பற்றிய நுண்ணறிவுகளைப் பெறவும், சில பாடங்கள் அல்லது தலைப்புகளைக் கற்கும்போது ஏற்படும் சிரமங்களை அடையாளம் காணவும், அவர்களின் பள்ளி வாழ்க்கை மற்றும் அவர்களின் வகுப்புத் தோழர்களுடனான உறவுகள் பற்றிய ஒரு புரிதலைப் பெறவும் வாய்ப்பாக அமைந்தது."[17]

மேலும், தற்போது சிங்கப்பூர் போன்ற நாடுகளில் தொற்றுநோய்ப் பரவலின் தாக்கம் கட்டுப்பாட்டுக்குள் வரத்தொடங்கிய நிலையில் பள்ளிகள், கல்லூரிகள் திறக்கப்பட்டன. ஆயினும், சமூக இடைவெளிகள், சுகாதாரம் போன்றவற்றில் கட்டாயப் பொதுநடைமுறைகள் பின்பற்றப்பட்டுக் கல்வித்துறையில் இயல்புநிலை திரும்பியது. குறிப்பிட்ட எண்ணிக்கையிலான மாணவர்களைக்கொண்ட வகுப்பறை, மின்னியல் வழியிலான பாடங்களை உள்ளடக்கிய 'கலவைமுறைக் கற்றல்' அணுகுமுறையின் செல்வாக்கு அதிகரிக்கத் தொடங்கியது. அதனடிப்படையில், 'வீட்டிலிருந்து நான்கு முதல் ஐந்து மணி நேரம் மாணவர்கள் பாடம் பயில்வர். அதில் இரண்டு மணி நேரம் மின்னிலக்கக் கருவிகளைப் பயன்படுத்துவர். பாதுகாப்பான இடைவெளி என்னும் நடைமுறையை ஆதரிக்கும் வண்ணம் பள்ளிகள் படிப்படியாகப் பலதரப்பட்ட கற்றல் முறைகளுக்கு மாறியுள்ளன. அதன் தொடக்கமாக வாரம் ஒரு நாள் இல்லம் சார்ந்த கற்றல் அறிமுகம் செய்யப்படுகிறது' என்று சிங்கப்பூர் கல்வி அமைச்சு அறிவித்தது[18]. மேலும், 'கல்வியிட வளாகத்திற்கு வெளியிலான கற்றல்' (OCL-Off Campus Learning) நடைமுறையும் புழக்கத்தில் வந்துள்ளது. இவை யாவும் தொழில்நுட்பம் சார்ந்தமையும் மின்னியல் கற்றலாகவே அமைகிறது. இவ்வாறு, மரபுசார் கற்றலும் தொழில்நுட்பம் சாரந்தமையும் கற்றல்-கற்பித்தலும் இணைந்தமையும் கலவைமுறைக் கற்றல் அணுகுமுறை நடைமுறையாக்கம் பெற்றுக் கல்வித்துறையின் மேம்பாட்டுக்கு வழிவகுத்துள்ளது.

கலவைமுறைக் கற்றல்

'கலவைமுறைக் கற்றல்' என்பது கல்விக்கான அணுகுமுறையாகும். இது புதுமையான அணுகுமுறை அல்ல; இக்கால வகுப்பறையில் நாள்தோறும் செய்துகொண்டிருக்கும் நடைமுறைதாம். ஆயினும் சற்று மாறுபட்ட செயல்பாட்டைக் கொண்டுள்ளது. "கலவைமுறைக் கற்றல் என்பது வெவ்வேறு கற்றல் சூழல்களின் கலவையாகும். இது பாரம்பரியக் கல்வி முறைகள் மற்றும் புதிய கற்றல் தொழில்நுட்பங்களின் கலவையை உள்ளடக்கியது."[19] என்பர். எனவே, மின்னியல் வழியிலான முறைகளையும் பாரம்பரியமான மரபுசார்ந்த

[17] https://www.nie.edu.sg/teacher-education/useful-resources-for-HBL-WFH-EdCo/tips-for-faculty-teachers-for-home-based-learning-(hbl)
[18] https://www.tamilmurasu.com.sg/singapore/story20200328-41982.html_ 28 March 2020
[19] https://singteach.nie.edu.sg/issue26-teachered-2/

நேருக்கு நேர் கற்பிக்கும் முறைகளையும் ஒருங்கிணைத்துச் செல்லும் அணுகுமுறைதான் கலவைமுறைக் கற்றல் என்பது தெளிவு. கலவைமுறைக் கற்றல் ஆசிரியர் மற்றும் மாணவர் இருவரின் பங்கேற்பையும் உள்ளடக்கிய மரபுவழிப்பட்ட கற்பித்தல் முறையுடன் ஒத்துச் செல்கிறது. ஆயினும், தொழில்நுட்பம் சார்ந்து இயங்கும் வேளையில் நேரம், இடம், செயற்படுத்தும் முறை, வேகம் போன்றவற்றில் மாறுபட்ட போக்கினைக் கொண்டுள்ளது. ஆசிரியர், மாணவர் நேரடிச்சந்திப்பு, இடைவினையாடல் போன்ற சில கட்டுப்பாடுகளும் இதில் உள்ளன. ஆசிரியர்கள் மாணவர்களுடன், நேருக்கு நேர் சந்திக்கும் வகுப்பறை நடைமுறைகள், பாடப்பொருள் வளமைகள் மற்றும் பயிற்சிகள் மற்றும் மதிப்பீடுகள் போன்ற கூறுகள் தொழில்நுட்பம் சார்ந்து கணினியுடனோ கைத்தொலைபேசியுடனோ புதுவகை நடவடிக்கைகளுடன் இணைக்கப்பட்டுச் செயல்படுத்தப்படுகின்றன. தொழில்நுட்பத்தின் செயல்பாட்டில் மாணவர்கள் அல்லது பங்கேற்பாளர்கள் அனைவருமே சிலவேளைகளில் தொலைவு என்ற ஒன்றினால் பிரிந்து நிற்கும் சூழல் எழுகிறது. ஆயினும், மாணவர்களின் கற்றல் அடைவுநிலை கலவைமுறைக் கற்றலில் அதிகம் என்பது அறிஞர்களின் கருத்து. இக்கலவை முறைக் கற்றல் தொழில்முறை மேம்பாடு மற்றும் பணித்திறன் பயிற்சி சார்ந்த அமைப்புகளிலும் இன்று பெருமளவில் பயன்படுத்தப்பட்டு வருகிறது.

கலவைமுறைக் கற்றலில் தொழில்நுட்பம்

கலவைமுறைக் கற்றல் சூழல் சார்ந்து அமைவது. அது அகச் சூழலாகவோ புறச்சூழலாகவோ அமையும். அகச்சூழல் என்பது மரபுசார் கற்றலை மையமிட்டது. வகுப்பறைக்கு உள்ளே நிகழ்பவற்றையும் நிகழ்த்துபவற்றையும் இது முன்னிருத்தும். ஆனால், புறச்சூழல் என்பது 'சார்ஸ்', 'கொவிட்-19' மற்றும் பேரிடர்க் காலங்களில் கற்றல்-கற்பித்தலில் தடைகள் ஏற்படும் சூழலில் அத்தடையை நீக்கும் முகமாகத் தொழில்நுட்பத்தைத் தளமாகக்கொண்டு இணைய வழியிலான மின்னியல் பாடங்களைச் சார்ந்திருக்கும். இக்கற்றல்முறை முகம் தெரியாமல் நடக்கும் பாடப்பொருளையும் வளங்களையும் நேருக்கு நேர் பார்த்தும் பயன்படுத்தியும் கற்க மாணவர்களைத் தூண்டுகிறது. மேலும், வகுப்பறையில் கற்கும் பாடத்தை மாணவர்கள் வகுப்பறைக்கு வெளியே உள்ளவற்றோடு பொருத்திப் பார்க்கவும் கற்பிக்கிறது. இதன்மூலம் ஆசிரியரின் செயல்பாடுகள் மாணவர்களின் படைப்பாற்றலோடு இணைந்து கலவைமுறைக் கற்றலாய் வெளிப்படுகிறது.

கலவைமுறைக் கற்றலில் ஈடுபடும் மாணவர்கள் தங்கள் அனுபவ அறிவைக்கொண்டு பாடத்தைப் புரிந்துகொள்வதோடு, தங்கள் முன்னறிவைத் தொழில்நுட்ப அறிவோடு இணைத்தும் பார்க்கின்றனர். மேலும், வகுப்பறையில் கற்கும் ஒன்றை வெளியில் நடக்கும் ஒன்றோடு பொருத்திப் பார்க்கவும் தொழில்நுட்பம் உதவுகிறது. அதோடு, மாணவர்கள் வேறு வேறு கற்றல் தளங்களுக்குச் சென்று கற்பதற்கும் தான் விரும்புவதை விரும்பும் நேரத்திலோ

விரும்பும் விதத்திலோ கற்பதற்கும் தொழில்நுட்பம் சார்ந்த இவ்வணுகுமுறை பெரிதும் பயன்படுகிறது[20].

கலவைமுறைக் கற்றலில் தொழில்நுட்பத்தின் செல்வாக்கு

கலவைமுறைக் கற்றலில் தொழில்நுட்ப வளங்களின் பயன்பாடு அதிகம். இதற்குக் காரணம், அதன் பயன்பாட்டுத்தன்மை (Usability) மிகவும் எளிதாக இருக்கிறது. புத்தகங்களைச் சுமக்கவும் பயிற்சி ஏடுகளை விலைகொடுத்து வாங்கவும் வேண்டிய கட்டாயமில்லை. மேலும், எந்த இடத்திலும், எந்த நேரத்திலும் தொழில்நுட்பத்தின் வழியாகக் கற்க இயலும். தரவுகள் மற்றும் பயிற்சிகளைச் சேமிக்கவும், மீண்டும் பயன்படுத்தவும் இயல்வதோடு வீட்டின் இடப்பரப்பு ஆக்கிரமிப்பும் இதிலில்லை. அதேவேளையில் செயல்பாட்டுத்தன்மையிலும் (Functionality) சிறப்பாக உள்ளது. உடனடியாகவும் விரைவாகவும் செயல்படும் தன்மை தொழில்நுட்பத்திற்கு உண்டு. எனவே, தேவையான தகவலைப் பெற நூலகத்தை நாடிச்செல்ல வேண்டிய கட்டாயமில்லை. உடனுக்குடன் தரவிறக்கம் செய்து பயன்படுத்துவதற்கு வாய்ப்பளிப்பதோடு நேரச்செலவையும் பொருட்செலவையும் குறைக்கிறது.

தனிநபர் உரிமை (Privacy) பேணப்படுவதும் இக்கற்றல் முறையின் தனிச்சிறப்புகளுள் ஒன்று. தொழில்நுட்பத்தால் ஒவ்வொரு மாணவரும் தனித்தனியே தன்னுடைய ஆசிரியரோடு இணைக்கப்படுகிறார். இதனால், சகமாணவர்களோடு வீணாகப் போட்டியிடுவதற்கும் ஒப்பீடு செய்துகொள்வதற்கும் வாய்ப்பில்லை. தன்னுடைய கற்றல் அடைவுநிலையை அந்த மாணவனும் ஆசிரியரும் மட்டுமே அறிவர். மேலும், தொழில்நுட்ம் வழிக் கற்றலில் நெகிழ்வுத்தன்மை (Flexibility) உண்டு. நாளொன்றுக்குச் செலவிடும் கால அளவிலும் இடவரையறையிலும் நெகிழ்வுத்தன்மை இருக்கிறது. மேலும் மரபுசார் வகுப்பறையில் பயன்படுத்தப்படும் கூடிக்கற்றல் (Collaborative Learning) அணுகுமுறையும் இங்குப் பயன்படுத்தப்படுகிறது. மாணவர்கள் ஒவ்வொருவரும் தனித்தனி இடங்களிலோ தொழில்நுட்பச் சாதனங்களிலோ இருந்தாலும் அவர்கள் யாவரும் மெய்நிகர் வழியில் உரையாடவும், கருத்துப் பரிமாற்றம் செய்யவும் வாய்ப்பளிக்கிறது. [21] மொத்தத்தில் மாணவர்-ஆசிரியர் இருவரும் இணைந்து செயலாற்றும் ஒரு கூட்டுத்தளமாக இக்கலவைமுறைக் கற்றல் அமைகிறது.

கலவைமுறைக் கற்றலுக்கான தொழில்நுட்ப வளங்களின் பன்முகம்

இன்றைய மின்னிலக்க உலகில் புதுப்புதுத் தொழில்நுட்பங்கள் நாளுக்குநாள் பிறப்பெடுத்துக்கொண்டே இருக்கின்றன. இவை, தேவை கருதியும் பயன்பாட்டு எளிமை கருதியும் கண்டுபிடிக்கப்பட்டுப் பயனாக்கம் பெறுகின்றன. தொழில்நுட்பப் பயனாளர்களாக மாறியிருக்கும் இன்றைய மாணவர் சமுதாயத்தைத் தொழில்நுட்ப உலகோடு பயணிக்கச் செய்யும் கடப்பாடு

[20] https://iteachwell.blogspot.com/2020/?m=0
[21] https://iteachwell.blogspot.com/2020/?m=0_Mobile learning

கல்வித்துறைக்கு நேர்ந்துள்ளது. அவ்வகையில் தொழில்நுட்பம் சார்ந்த கற்றல் அதிகம் வலியுறுத்தப்படுகின்றது. மேலும், பருவநிலை மாற்றம், பேரிடர் நிகழ்வுகள், தொற்றுநோய்ப் பரவல் போன்ற காரணிகள் அவ்வப்போது கல்விக்குக் குறுக்கே நிற்கும் சூழலில் தொழில்நுட்பத்தின் ஆளுமை கற்றல்-கற்பித்தலில் புகவேண்டிய கட்டாயநிலை ஏற்படுகிறது. அவ்வகையில் பல்வேறு தொழில்நுட்ப வளங்கள் கலவைமுறைக் கற்றலில் பயன்படுத்தப்படுகின்றன. இவ்வளங்களால் கட்புலன், செவிப்புலன் வழியாக அறிவு புகட்டப்பட்டு, கற்றலின் ஆழம் வலுப்படுகிறது. கடினமான பாடங்களைக்கூட எளிதில் புரிந்துகொள்ளவும், கருத்தாடலையும் விவாதங்களையும் ஊக்குவித்துச் (simulations) சிந்தனைத் திறனை மேம்படுத்தவும் இவை உதவுகின்றன. தொழில்நுட்பம் சார்ந்தமையும் வளங்கள் பள்ளிசார் தளங்கள், வகுப்பறை வளங்கள், இணைய வகுப்பறைகள், கற்றல் வளமையகம், மதிப்பீட்டுக் கருவிகள் என்று பன்முகப் பயன்பாட்டோடு கற்றல்-கற்பித்தலை வலுவூட்டி ஆக்கநிலைக்கு இட்டுச்செல்கின்றன.

கல்விநிலையங்களுக்குரிய வளங்கள்

பொதுவாகக் கல்விநிலையங்கள் தங்களுக்கெனத் தனிப்பட்ட தொழில்நுட்பம் சார்ந்த கற்றல் தளத்தைப் பயன்படுத்துவதைச் சிங்கப்பூர்ப் பள்ளிகளும் உயர்கல்வி நிலையங்களும் நடைமுறைப்படுத்தி வருகின்றன. கற்றல் மேலாண்மைத் தளங்களான (LMS-Learning Management System), Blackboard, Leo 2.0 போன்ற வளங்களைப் பயன்படுத்துவதோடு, இருவழித் தொடர்புத்தளம் (iMTL), மாணவர் கற்றல் தளம் (SLS-Students Learning Space) போன்ற கல்வி அமைச்சின் கற்றல் தளங்களையும் அவற்றின் வளங்களையும் பயன்படுத்துகின்றனர். இவையாவும் தொழில்நுட்பம் சார்ந்தனவாகவும் இருவழிக் கருத்துப் பரிமாற்றத்திற்கு (பேச்சு, எழுத்து) இடமளிப்பனவாகவும் அமைகின்றன. இன்று பெரும்பாலான பாடநடவடிக்கைகளும் பயிற்சிகளும் இவ்வளங்களை அடிப்படையாகக்கொண்டே அமைகின்றன. மேலும், Opal 2.0 எனப்படும் கற்றல் தளம் ஆசிரியர்க்கு உதவும் பாடவளமைகளை உள்ளடக்கியுள்ளது என்பது குறிப்பிடத் தக்கது.

ஆர்வமூட்டும் வகுப்பறை வளங்கள்

இன்று பெரும்பான்மையான கல்வி நிலையங்கள் கற்றல்-கற்பித்தலில் தொழில்நுட்பப் பயன்பாடு இருப்பதை ஆதரிக்கின்றன. அவ்வகையில் பெரும்பாலான வகுப்பறை நடவடிக்கைகளில் தொழில்நுட்ப வளங்களின் பயன்பாடு மிகுந்துள்ளது. அதனடிப்படையில் மாணவர்களை ஈடுபாடுமிக்க அனுபவழிக் கற்றல் (*Engaged* and experience *learning)* மற்றும் மகிழ்வூட்டும் கற்றல் (*Joy of learning)* அணுகுமுறைகளுக்கு இட்டுச்செல்லும் பல்வேறு தொழில்நுட்ப வளங்களின் பயன்பாடு இன்று வகுப்பறையை ஆக்கிரமித்துள்ளன. அவ்வகையில் Kahoot, Mentimeter, Slido, Padlet போன்ற வளங்களைக் குறிப்பிடலாம். இவை பாடப்பொருள் பற்றிய புரிதலைச் சோதிப்பதற்கும் நேரடி

வாக்கெடுப்பு நிகழ்த்துவதற்கும் பயன்படுத்தப்படுகிறது. மாணவர்கள் உற்சாகத்துடன் இவற்றில் பங்கெடுப்பது கற்றலை ஈடுபாடுமிக்கதாக ஆக்குகிறது.

இணைய வகுப்பறைகள்

மரபுவழிப்பட்ட வகுப்பறைச் சூழலைத் தாண்டி, இன்று இணைய வகுப்பறைகள் கற்றல்-கற்பித்தல் தளத்தினுள் புகுந்துவிட்டன. அதனடிப்படையில் கூகுள் வகுப்பறை, ஜூம், ஸ்கைப், கூகுள் மீட், ஸ்கைப் போன்ற தொழில்நுட்பம் சார்ந்த வளங்களைக் கொண்டமையும் மின்னியல் வகுப்பறைகள் பயன்பாட்டில் உள்ளன. இன்றைய 'கொவிட்-19' நெருக்கடிச் சூழலில் இவை ஒருவிதப் புது உத்வேகத்தோடு செயல்படுகின்றன. மாணவர் கற்றலுக்கு மட்டுமல்லாது இணைய வழிக் கருத்தரங்கம், பயிலரங்கம் போன்றவற்றை நிகழ்த்தவும் இவை உதவுகின்றன. இணைய வகுப்பறைகள் மரபுசார் வகுப்பறைச் செயல்பாடுகளை உள்ளடக்கி, அதனின் மற்றொரு பரிணாமத்தையும் நடைமுறை உலகில் சாத்தியமாக்கியுள்ளது என்பது துணிபு.

கற்றல் மற்றும் மதிப்பீட்டு வளங்கள்

கற்றல்-கற்பித்தலை வளமூட்டப் பல்வேறு தொழில்நுட்ப வளங்கள் பன்முகத் தன்மையுடன் பயன்பாட்டில் உள்ளன. அவற்றுள் கூகுள் நிறுவனத்தின் வளங்கள் குறிப்பிடத்தக்கவை. 'ஜீ சூட்' செயலிகள் பெரும்பாலும் வகுப்பறை சார்ந்த கற்றல் வளங்களாக உள்ளன. ஒருங்கிணைப்பு நடவடிக்கைகள், வகுப்பறை நிர்வாகம் போன்றவற்றுக்கும் இவை பயன்படுகின்றன. இதுபோல் MindMeister எனப்படும் தொழில்நுட்ப வளம் மனவரைபடம் வழிக் கற்றலுக்கு உதவுகிறது. இவ்விணைய வளங்கள் இலவசமாகவும் விலைக்கும் கிடைக்கின்றன. மேலும், தொழில்நுட்பம் சார்ந்த மதிப்பீட்டு வளங்களும் இன்று பயன்பாட்டில் உள்ளன. Adobe acrobat pro DC, Kami - PDF and Document Annotation, Classkick போன்ற இணைய வளங்கள் மாணவர்களின் மின்னியல் ஒப்படைப்புகளைத் திருத்தவும் மதிப்பீடு செய்யவும் பயன்படுகின்றன. 'கொவிட்-19' சூழலில் இல்லம் சார் கற்றல் நிகழ்ந்தபோது ஆசிரியர்களுக்குப் பக்கபலமாக இருந்து, கற்றல்-கற்பித்தலை வலுவூட்டியவை இத்தகு தொழில்நுட்பங்களே என்றால் அது மிகையாகாது.

முடிவுரை

தொழில்நுட்பத்தின் செல்வாக்குத் தமிழ்மொழி வளர்ச்சிக்கு வித்திட்டதோடு, கற்றல் கற்பித்தல் மேம்பாட்டிற்கும் உதவிக்கரம் நீட்டிவருகின்றது. அதனடிப்படையில் உலகை அச்சுறுத்திக்கொண்டிருக்கும் 'கொவிட்-19' நெருக்கடிச் சூழலில் கல்விநிலையங்கள் இல்லம்சார் கற்றலை நடைமுறைப்படுத்தின. அதனைத்தொடர்ந்து கலவைமுறைக் கற்றல் நடைமுக்கு வந்தது. தொழில்நுட்பத்தின் மேலாதிக்கம் அதிகரிக்கத் தொடங்கிய இச்சூழலில் வகுப்பறைச் செயல்பாடாகவும் மதிப்பீட்டுக் கருவியாகவும் கற்றல்-கற்பித்தல்

நெறியாகவும் தொழில்நுட்ப வளங்கள் கலவைமுறைக் கற்றலில் பயன்படுத்தப்படுகின்றன. இவ்வளங்கள் கற்றல்-கற்பித்தலை இடையறாது செயல்படுத்தவும் அதை அர்த்தமுள்ள ஒன்றாகவும் ஆக்கியுள்ளது என்பது கண்கூடு.

ஜ நிறைவு ஜ

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

மாத்திரை பார்வையில் குறள்

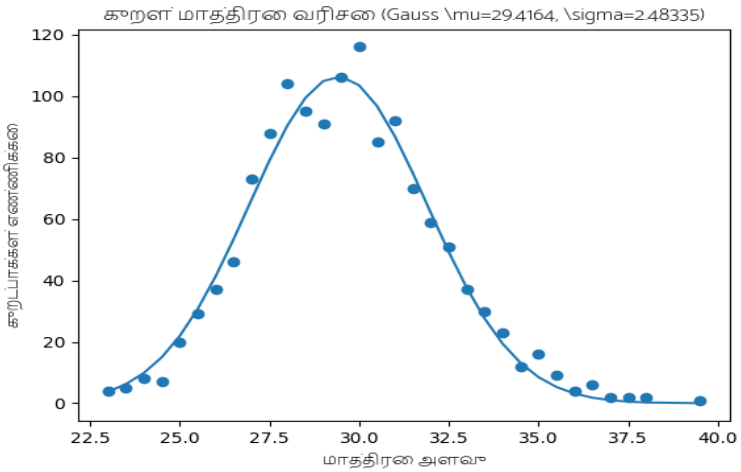
ஆசிரியர்: பரதன் தியாகலிங்கம், முத்து அண்ணாமலை
தொடர்பு மின் அஞ்சல்: ezhillang@gmail.com

திருக்குறளின் [1] 1330 குறட்பாக்களை மாத்திரை பார்வையில் கணினிவழியாக இயல்மொழி ஆய்வு செய்தால் என்ன கிடைக்கும்? திருக்குறளை மாத்திரை மதிப்பின் வாயிலாக வரிசைப்படுத்திப் பார்த்தால் என்ன கிடைக்கும் ? ஏதேனும் புதிய புரிதல் உண்டாகிறதா? பார்க்கலாம் வாருங்கள்.

செய்முறை – அல்கோரிதம்

குறளின் மாத்திரை அளவு என்பது குறளின் உள்ள அனைத்து சீர்பிரிக்காத சொற்களின் தனி மாத்திரை அளவுகளின் சமன்பாடு என்று கொள்ளலாம். இது நமது ஆய்வின் முன்கூட்டிய புரிதல்.

முதலில் இதற்கு ஒரு தமிழில் உள்ள மாத்திரை விதிகளை கணிக்கும் சார்பு தேவைப்படுகிறது. இதனை open-tamil [2] தொகுப்பில் 'tamil.utf8.total_maththirai' என்ற நிரல்துண்டு வழங்குகிறது. மேலும் குறட்பக்களை 'kural.Thirukkural().get_kural_no()' என்பதிலிருந்து பெரலாம். இரண்டினையும் சேர்த்து ஒரு சிரிய கோவ்சியன் வளையம் பொருத்தலுடன் இணைத்துப்பார்த்தால் இப்படி தெரிகிறது; இதன் மூல நிரல் kural_mathirai.py என்பதில் காணலாம் அல்லது பின் இணைப்பு 1-இல் காணலாம். மேலும் இதனை ஒரு சராசரி கௌசியன் வளைவில் பொருத்தினால் அதன் நடுநிலை மாத்திரை அளவு 29.5 ஆகவும், வேறுபாடுகள் அளவு +/-2.5 என்றும் படம் 1-இல் கண்டபடி வந்தது.



படம் 1: திருக்குறள் குறட்பாக்களை மாத்திரை வடிவில் ஒரு கௌசியன் வளைவில் பொருத்திம் ஆகிறது.

விடைகள்

1. திருக்குறளில் உள்ள சராசரி குறட்பாவின் மாத்திரை அளவு μ ~ 29.5. இதன் மாற்றமளவு σ ~ 2.5
2. மாத்திரை பார்வையில் திருக்குறள் ஏரக்குறைய கௌசியன் பரப்பை போல் அமைந்துள்ளது
3. திருக்குறள் மாத்திரை வடிவிலும் கூட அழகிய சீர்மை கொண்டதாக மிகவும் கோர்வையுடன் அமைந்தது.
4. குறைந்த அளவு நமாத்திரை நீளம் (23) கொண்ட குறளானவை குறள் எண்கள், 391, 426, 483, 786

   “கற்க கசடறக் கற்பவை கற்றபின்

   நிற்க அதற்குத் தக.” குறள் **391**.

5. அதிக அளவு நீளமான மாத்திரை (37.5) கொண்ட குறளானது குறள் வரிகள்,

   ”காணாதான் காட்டுவான் தான்காணான் காணாதான்

   கண்டானாம் தான்கண்ட வாறு.” குறள் **849**.

திருக்குறள் மாத்திரை வரிசை ஒத்திய குறட்பா எண்ணிக்கை

மத்திரை அளவு	குறள் எண்ணிக்கை	குறள் எண்(கள்)
23	4	234391, 426, 483, 786
23.5	5	67, 77, 366, 637, 979
24	8	108, 485, 961, 965, 1042, 1048, 1277, 1304
24.5	7	304, 467, 602, 652, 1118, 1322, 1324
25	20	133, 193, 331,360, 412,477, 546, 559, 576, 592, 616, 771, 796, 846, 947,1052, 1116, 1239, 1240, 1289
25.5	29	96, 339, 347, 350, 373, 400, 405, 439, 444, 569,

		625, 669, 742, 764, 841, 937, 944, 978, 980, 1045, 1072, 1085, 1102, 1108, 1176, 1203, 1258, 1318,1329
26	37	[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]
26.5	46	[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]
27	73	[80, 105, 109, 119, 120, 121, 172, 173, 174, 202, 224, 231, 283, 284, 292, 321, 337, 341, 349, 374, 380, 399, 411, 428, 429, 438, 447, 489, 505, 508, 512, 520, 552, 562, 570, 578, 599, 604, 651, 654, 661, 693, 698, 729, 747, 770, 772, 778, 790, 803, 805, 807, 817, 818, 822, 832, 851, 863, 887, 889, 963, 994, 1019, 1041, 1069, 1107, 1120, 1122, 1159, 1173, 1300, 1321, 1326]
27.5	88	[21, 34, 39, 64, 73, 87, 95, 98, 111, 115, 124, 131, 140, 157, 205, 207, 208, 210, 215, 239, 261, 270, 272, 280, 293, 309, 315, 336, 388, 394, 419, 431, 457, 465, 468, 469, 482, 491, 494, 495, 542, 543, 545, 554, 558, 574, 575, 581, 629, 653, 657, 679, 690, 705, 739, 759, 788, 811, 821, 823, 835, 876, 878, 882, 883, 904, 918, 949, 1004, 1023, 1039, 1063, 1091, 1105, 1109, 1117, 1138, 1144, 1156, 1165, 1166, 1226, 1257, 1266, 1273, 1281, 1283, 1301]
28	104	13, 14, 60, 63, 75, 85, 89, 101, 103, 123, 145, 158, 167, 186, 191, 195, 213, 218, 230, 234, 302, 327, 369, 408, 410, 416, 425, 434, 463, 464, 472, 475, 478, 500, 504, 521, 522, 535, 549, 563, 580, 595, 596, 609, 618, 620, 621, 636, 664, 677, 703, 741, 748, 752, 767, 793, 797, 802, 816, 844, 873, 880, 884, 917, 925, 943, 984, 1003, 1011, 1013, 1031, 1040, 1043, 1051, 1080, 1086, 1088, 1092, 1096, 1099, 1100, 1114, 1135, 1137, 1141, 1146,

		1153, 1157, 1177, 1178, 1184, 1199, 1201, 1208, 1210, 1216, 1221, 1259, 1264, 1295, 1306, 1307, 1327, 1330
28.5	95	29, 68, 102, 104, 112, 114, 116, 132, 137, 153, 203, 223, 227, 238, 244, 255, 265, 267, 268, 274, 277, 295, 303, 316, 333, 342, 372, 381, 385, 392, 393, 398, 401, 403, 407, 437, 493, 506, 511, 517, 553, 577, 591, 601, 605, 631, 655, 671, 674, 696, 710, 723, 727, 740, 744, 749, 757, 763, 781, 809, 819, 829, 839, 852, 871, 941, 986, 996, 1025, 1050, 1061, 1062, 1073, 1093, 1098, 1119, 1142, 1152, 1164, 1170, 1196, 1209, 1213, 1223, 1249, 1250, 1268, 1280, 1287, 1290, 1298, 1308, 1310, 1315, 1328
29	91	23, 26, 59, 76, 99, 113, 169, 181, 184, 197, 214, 233, 237, 254, 262, 264, 289, 313, 329, 334, 335, 338, 384, 427, 503, 539, 544, 547, 588, 590, 597, 607, 619, 630, 634, 638, 672, 707, 709, 712, 728, 754, 779, 785, 787, 791, 810, 812, 813, 826, 830, 837, 848, 853, 879, 888, 898, 905, 920, 942, 955, 971, 1014, 1021, 1034, 1068, 1084, 1089, 1110, 1121, 1126, 1139, 1143, 1147, 1149, 1151, 1172, 1193, 1200, 1211, 1231, 1237, 1245, 1247, 1251, 1253, 1271, 1276, 1292, 1299, 1305
29.5	106	[9, 33, 53, 55, 61, 66, 71, 79, 100, 136, 150, 154, 171, 177, 182, 192, 216, 226, 259, 279, 288, 290, 306, 310, 323, 344, 352, 363, 371, 375, 376, 377, 378, 404, 414, 418, 440, 448, 449, 455, 486, 497, 501, 527, 532, 585, 589, 593, 603, 641, 644, 650, 688, 697, 704, 726, 731, 734, 736, 743, 746, 751, 761, 765, 782, 801, 814, 815, 831, 836, 842, 850, 870, 872, 899, 922, 933, 945, 953, 958, 970, 974, 982, 997, 1000, 1038, 1053, 1065, 1067, 1095, 1111, 1113, 1128, 1161, 1171, 1219, 1228, 1234, 1235, 1236, 1244, 1252, 1254, 1261, 1284, 1297]
30	116	[19, 22, 48, 49, 57, 78, 97, 117, 122, 125, 129, 142, 151, 178, 180, 201, 217, 220, 241, 245, 250, 257, 260, 269, 273, 276, 286, 300, 332, 348, 353, 362, 365, 370, 420, 421, 432, 435, 436, 443, 470, 474, 479, 490, 498, 499, 507, 509, 524, 529, 534,

		556, 565, 568, 610, 613, 626, 633, 640, 647, 659, 663, 676, 684, 714, 725, 750, 775, 776, 784, 798, 799, 800, 828, 864, 869, 885, 886, 890, 891, 892, 893, 902, 907, 916, 946, 948, 951, 962, 999, 1015, 1016, 1044, 1047, 1049, 1055, 1056, 1074, 1076, 1082, 1124, 1125, 1127, 1155, 1175, 1179, 1180, 1183, 1204, 1207, 1217, 1230, 1243, 1263, 1282, 1319]
30.5	85	[10, 32, 58, 62, 83, 84, 88, 92, 139, 179, 200, 209, 235, 243, 251, 287, 291, 294, 297, 299, 320, 322, 346, 356, 367, 386, 396, 415, 430, 441, 442, 458, 459, 519, 525, 537, 557, 567, 624, 645, 656, 665, 678, 701, 717, 724, 773, 783, 789, 855, 859, 895, 906, 912, 915, 977, 992, 995, 1012, 1024, 1029, 1059, 1078, 1087, 1090, 1094, 1103, 1106, 1115, 1129, 1140, 1145, 1163, 1214, 1225, 1262, 1265, 1274, 1275, 1291, 1293, 1303, 1313, 1320, 1323]
31	92	[16, 17, 18, 30, 37, 46, 50, 70, 72, 74, 93, 106, 135, 138, 144, 146, 160, 161, 190, 198, 204, 211, 222, 307, 319, 351, 358, 382, 383, 390, 406, 422, 460, 471, 473, 496, 523, 528, 538, 555, 561, 566, 587, 600, 612, 615, 648, 670, 702, 716, 721, 737, 753, 760, 768, 795, 820, 843, 857, 862, 866, 874, 903, 909, 985, 987, 989, 991, 1018, 1020, 1030, 1037, 1054, 1060, 1077, 1123, 1130, 1136, 1154, 1158, 1162, 1169, 1185, 1188, 1189, 1198, 1206, 1218, 1260, 1278, 1288, 1314]
31.5	70	[3, 4, 11, 36, 41, 65, 69, 81, 110, 189, 242, 258, 263, 275, 278, 324, 354, 355, 359, 361, 379, 413, 417, 476, 481, 510, 536, 541, 573, 579, 586, 614, 617, 649, 658, 680, 694, 718, 722, 755, 766, 794, 804, 845, 858, 860, 867, 913, 914, 923, 926, 928, 936, 952, 956, 957, 976, 990, 998, 1022, 1026, 1027, 1033, 1035, 1075, 1104, 1167, 1182, 1248, 1255]
32	59	[12, 44, 52, 128, 143, 156, 164, 165, 185, 188, 228, 253, 298, 311, 312, 345, 357, 389, 395, 433, 445, 450, 513, 530, 560, 564, 606, 642, 673, 686, 711, 730, 732, 735, 758, 762, 777, 792, 825, 833,

		840, 881, 919, 968, 1036, 1057, 1058, 1081, 1112, 1160, 1174, 1197, 1212, 1215, 1272, 1294, 1311, 1316, 1325]
32.5	51	[2, 20, 24, 31, 35, 47, 82, 86, 118, 147, 148, 159, 162, 176, 187, 271, 281, 282, 328, 446, 502, 514, 526, 533, 550, 582, 583, 662, 683, 699, 719, 720, 827, 861, 865, 908, 921, 927, 969, 981, 1097, 1150, 1181, 1186, 1187, 1191, 1194, 1195, 1241, 1286, 1312]
33	37	[8, 155, 170, 183, 206, 212, 225, 248, 249, 318, 325, 368, 402, 462, 492, 548, 627, 646, 675, 685, 756, 780, 808, 834, 911, 939, 1046, 1070, 1083, 1101, 1148, 1190, 1192, 1269, 1270, 1285, 1317]
33.5	30	[27, 194, 219, 221, 305, 317, 343, 456, 461, 632, 643, 667, 689, 806, 896, 900, 901, 930, 932, 959, 966, 972, 993, 1001, 1005, 1009, 1133, 1205, 1220, 1267]
34	23	[38, 51, 56, 126, 134, 166, 252, 266, 330, 453, 487, 660, 681, 682, 695, 897, 910, 938, 950, 960, 967, 1007, 1032]
34.5	12	[5, 42, 130, 149, 232, 387, 424, 733, 931, 983, 1006, 1079]
35	16	91, 94, 141, 199, 246, 247, 423, 452, 635, 691, 875, 924, 973, 1002, 1017, 1064
35.5	9	25, 163, 285, 301, 515, 584, 868, 894, 929
36	4	7, 409, 713, 745
36.5		127, 240, 326, 551, 1202, 1246
37	2	15, 940
37.5	2	43, 397
38.0	2	6, 256
39.5	1	849

```
#!/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'குறட்பாக்கள் எண்ணிக்கை',{'fontname':'Catamaran'})
    plt.xlabel(u'மாத்திரை அளவு',{'fontname':'Catamaran'}) #Arial Unicode MS'})

    # 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)

    # Get the fitted curve
    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]))
```

மேற்கோள்

1. திருக்குறள், திருவள்ளுவர்.

2. ஓப்பன் தமிழ் வெளியீடு வரிசை எண்: 0.97, (ஜூலை, 2020). https://pypi.org/project/Open-Tamil/

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

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 (வாய்பாடு) are used in grade school (K-12) to commit basic mathematical facts to memory and take a textual form as “இரண்டு நான்கு எட்டு", meaning 2 x 4 = 8. Ability to generate numbers in word-forms enables us to generate these multiplication tables ad-hoc. [4], [5].

Fig. 1. Multiplication table for 6 created using this numeral generation algorithm in [4].

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 100,0000.
    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

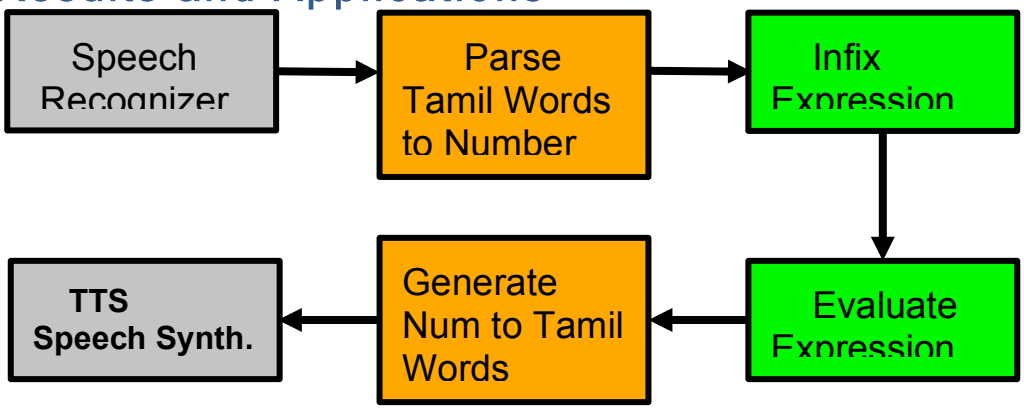


Fig. 2. Voice User-Interface for Tamil Spoken Calculator.

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

Input Expression	Parsed Expression	Evaluated Result	Output Result
ஒன்று கூட்டல் ஒன்று	1 + 1	2	இரண்டு
ஒன்று கூட்டல் இரண்டு பெருக்கல் பத்து	1 + 2 * 10	21	இருபத்தொன்று
ஓர் ஆயிரம் கழித்தல் ஐந்து பெருக்கல் (ஒன்பது கூட்டல் ஒன்று)	1000 - 5*(9 + 1)	950	தொள்ளாயிரத்து ஐம்பது
ஒரு இலட்சம் பெருக்கல் பத்து	100000*10	1000000	பத்து இலட்சம்

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 (ஒலினி)* [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 (அரை,கால், முக்கால், ... மற்றும் பல), ordinals, multiplication tables (வாய்பாடு), domain specific uses in colloquial spoken contexts e.g. “ஒன்றையும் பத்தையும் கூட்டு", "பத்தில் நாலை கழி," [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

1. Charles J. Swift, “Evaluating numbers expressed as strings of English words,” CACM Oct. (1960).
2. Vasu Renganathan, “Tamil Language in Context,” (2011).
3. (a) Patent - US 4,882,685 “Voice Activated Electronic Calculator,” van der Lely (1989).
   (b) Patent - US 2008.0312928A1 “Natural Language Speech Recognition Calculator,” Goebel, Shivanna (2008).
4. (a) Tamilpesu.us - Open-Tamil Indic Computing Platform; http://tamilpesu.us (accessed June 5th 2020).
   (b) Open-Tamil Git Repository - https://github.com/Ezhil-Language-Foundation/open-tamil (accessed June 2020).
5. S. Abuthahir, et-al “Growth and Evolution of Open-Tamil,” Tamil Internet Conference, Coimbatore, Tamilnadu. (2018).
6. W. H. Arden, “A Progressive Grammar of Common Tamil,” Soc. for Promoting Christian Knowledge (1910).
7. சிங்கப்பூர் சித்தார்த்தன், "இலகு தமிழில் இனிக்கும் தமிழ் இலக்கணம்," நர்மதா பதிப்பகம் (2017).
   Siddarthan, “A treatise and guide to learn Tamil Grammar,” Narmada Publications (2017).

A. Appendix - Python Source Code

A.1 Generating Tamil Number Words

```
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
isinstance(number,complex):
        raise Exception('num2tamilstr input has to be a 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'பூஜ்ஜியம்', u'ஒன்று', u'இரண்டு', u'மூன்று', u'நான்கு', u'ஐந்து', u'ஆறு', u'ஏழு',
u'எட்டு', u'ஒன்பது', u'பத்து') # 0-10
    units_suffix = (u'பூஜ்ஜியம்', u'தொன்று', u'திரண்டு', u'மூன்று', u'நான்கு', u'தைந்து',
u'தாறு', u'தேழு', u'தெட்டு', u'தொன்பது', u'பத்து') # 0-10
    units_suffix_nine = (u'பூஜ்ஜியம்', u'றொன்று', u'றிரண்டு', u'மூன்று', u'நான்கு',
u'றைந்து', u'றாறு', u'றேழு', u'றெட்டு', u'றொன்பது', u'பத்து') # 0-10
    tween = [1.0,2.0,5.0,6.0,7.0,8.0,9.0]
    teens = (u'பதினொன்று', u'பனிரண்டு', u'பதிமூன்று', u'பதினான்கு',
u'பதினைந்து',u'பதினாறு', u'பதினேழு', u'பதினெட்டு', u'பத்தொன்பது') # 11-19
    tens = (u'பத்து', u'இருபது', u'முப்பது', u'நாற்பது', u'ஐம்பது',u'அறுபது', u'எழுபது',
u'எண்பது', u'தொன்னூறு') # 10-90
    tens_full_prefix = (u'இருபத்து', u'முப்பத்து', u'நாற்பத்து', u'ஐம்பத்து', u'அறுபத்து',
u'எழுபத்து', u'எண்பத்து', u'தொன்னூற்று') # 10+-90+
    tens_prefix = (u'இருபத்', u'முப்பத்', u'நாற்பத்', u'ஐம்பத்', u'அறுபத்', u'எழுபத்',
u'எண்பத்', u'தொன்னூற்') # 10+-90+
```

```
    hundreds = ( u'நூறு', u'இருநூறு', u'முன்னூறு', u'நாநூறு',u'ஐநூறு', u'அறுநூறு',
u'எழுநூறு', u'எண்ணூறு', u'தொள்ளாயிரம்') #100 - 900
    hundreds_suffix = (u'நூற்றி', u'இருநூற்று', u'முன்னூற்று', u'நாநூற்று', u'ஐநூற்று',
u'அறுநூற்று', u'எழுநூற்று', u'எண்ணூற்று',u'தொள்ளாயிரத்து') #100+ - 900+

    one_thousand_prefix = u'ஓர்'
    thousands = (u'ஆயிரம்',u'ஆயிரத்து')

    one_prefix = u'ஒரு'
    lakh = (u'இலட்சம்',u'இலட்சத்து')
    crore = (u'கோடி',u'கோடியே')

    pulli = u'புள்ளி'
    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])
```

```
        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 teens[teen_number-1] +u' ' +
num2tamilstr(residue_number,filenames)
            else:
                return teens[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))
```

```
            numeral = hundreds_suffix[quotient_number-1]+u' '
        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
        return numeral+residue_numeral
    # number has to be zero
    filenames.append("units_0")
    return units[0]
```

A.2 Parsing Tamil Number Words

```
def tamilstr2num(tokens):
    """
        இயல்மொழி எண்பகுப்பாய்வு.
        numeral parser; convert numeral to number.
        e.g. ["இருநூற்று","நாற்பத்தைந்து"] => 245
    """
    if isinstance(tokens,str):
        tokens = re.split(SPACE,tokens)
    is_american_str = False
    has_decimal = False
```

```
    US_values = [1e12, 1e9, 1e6]
    US_placements = ['டிரில்லியன்', 'பில்லியன்','மில்லியன்']
    IN_values = [1e7,1e7,1e5,1e5]
    IN_placements = ('கோடி','கோடியே','இலட்சம்','இலட்சத்து',)
    for tok in tokens:
        if tok in US_placements:
            is_american_str =True
        elif tok == 'புள்ளி':
            has_decimal = True
    value = 0
    stack = list()
    if is_american_str:
        HIGHEST=('டிரில்லியன்',)
        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 = (u'பூஜ்ஜியம்', u'ஒன்று', u'இரண்டு', u'மூன்று', u'நான்கு', u'ஐந்து', u'ஆறு', u'ஏழு',
u'எட்டு', u'ஒன்பது', u'பத்து') # 0-10
    units_suffix = (u'பூஜ்ஜியம்', u'தொன்று', u'திரண்டு', u'மூன்று', u'நான்கு', u'தைந்து',
u'தாறு', u'தேழு', u'தெட்டு', u'தொன்பது', u'பத்து') # 0-10
    units_suffix_nine = (u'பூஜ்ஜியம்', u'றொன்று', u'றிரண்டு', u'மூன்று', u'நான்கு',
u'றைந்து', u'றாறு', u'றேழு', u'றெட்டு', u'றொன்பது', u'பத்து') # 0-10
    for _u in [units,units_suffix,units_suffix_nine]:
        for k,v in zip(_u,val_units):
            val_map[k]=v

    teens = (u'பதினொன்று', u'பனிரண்டு', u'பதிமூன்று', u'பதினான்கு',
u'பதினைந்து',u'பதினாறு', u'பதினேழு', u'பதினெட்டு', u'பத்தொன்பது') # 11-19
    for k,v in zip(teens,range(11,20)):
        val_map[k]=v
    tens = (u'பத்து', u'இருபது', u'முப்பது', u'நாற்பது', u'ஐம்பது',u'அறுபது', u'எழுபது',
u'எண்பது', u'தொன்னூறு') # 10-90
    for k,v in zip(tens,range(10,100,10)):
        val_map[k]=v
    tens_full_prefix = (u'இருபத்து', u'முப்பத்து', u'நாற்பத்து', u'ஐம்பத்து', u'அறுபத்து',
u'எழுபத்து', u'எண்பத்து', u'தொன்னூற்று') # 10+-90+
    for k,v in zip(tens_full_prefix,range(20,100,10)):
        val_map[k]=v
    tens_prefix = (u'இருபத்', u'முப்பத்', u'நாற்பத்', u'ஐம்பத்', u'அறுபத்', u'எழுபத்',
u'எண்பத்', u'தொன்னூற்') # 10+-90+
    for k,v in zip(tens_prefix,range(20,100,10)):
        val_map[k]=v
    hundreds = ( u'நூறு', u'இருநூறு', u'முன்னூறு', u'நாநூறு',u'ஐநூறு', u'அறுநூறு',
u'எழுநூறு', u'எண்ணூறு', u'தொள்ளாயிரம்') #100 - 900
    for k,v in zip(hundreds,range(100,1000,100)):
        val_map[k]=v
    hundreds_suffix = (u'நூற்றி', u'இருநூற்று', u'முன்னூற்று', u'நாநூற்று', u'ஐநூற்று',
u'அறுநூற்று', u'எழுநூற்று', u'எண்ணூற்று',u'தொள்ளாயிரத்து') #100+ - 900+
    for k,v in zip(hundreds_suffix,range(100,1000,100)):
        val_map[k]=v
    one_thousand_prefix = u'ஓர்'
    val_map[one_thousand_prefix] = 1.0
```

```
thousands = (u'ஆயிரம்',u'ஆயிரத்து')
val_map[thousands[0]] = 1000.0
val_map[thousands[1]] = 1000.0
one_prefix = u'ஒரு'
val_map[one_prefix] = 1.0
lakh = (u'இலட்சம்',u'இலட்சத்து')
val_map[lakh[0]] = 100000.0
val_map[lakh[1]] = 100000.0
crore = (u'கோடி',u'கோடியே')
val_map[crore[0]] = 10000000
val_map[crore[1]] = 10000000
pulli = u'புள்ளி'
val_map[pulli] = 0.0

mil = u'மில்லியன்'
val_map[mil] = 1000000.0
bil = u'பில்லியன்'
val_map[bil] = val_map[mil]*1e3
tril = u'டிரில்லியன்'
val_map[tril] = val_map[bil]*1e3
in_fractional_portion = False
multiplier = 1.0
value = 0.0



#["இருநூற்று","நாற்பத்தைந்து"] => 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
```

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

SymSpell and LSTM based Spell-Checkers for Tamil

Selvakumar Murugan, Tamil Arasan Bakthavatchalam, Malaikannan Sankarasubbu

Saama Technologies AI Research Lab

{selvakumar.murugan, t.arasan, malaikannan.sankarasubbu}@saama.com

Nov 15, 2020

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-mei* 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. *அகளவிரித்து → அகலவிரித்து, அனுகப்பட → அணுகப்பட, ஆளுனரால் அனுப்பி → ஆளுநரால் அனுப்பி, காலத்திர்க்கானவை → காலத்திற்கானவை*

Language Dependent Errors

Input engine issues: some input systems follow the scheme: *uyirmei* + *pulli* → *mei* and some follow scheme: *mei* + *uyir* → *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: காலை > காலையில் > காலையிலிருந்து > காலையிலிருந்தே > காலையிலிருந்தேயவன்.

Agglutination have brings in alphabets that are not in the combined words which are a common source of error in writing Tamil, e.g: இணைத்து பார்த்தேன் → இணைத்துப் பார்த்தேன்; இணைந்துப் படிப்பதற்கும் → இணைந்து படிப்பதற்கும். Notice that depending upon the grammatical context, the alphabet ப் appears or vanishes.

Some are not strict grammatical rules but guidelines to elucidate a cleaner context e.g: கிராமப் புறங்களில் இருந்து → கிராமப்புறங்களிலிருந்து.

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.

Corpus	Count
all_tamil_nouns/all_nouns.txt	181185
tharavukkanam/tamil-etymological-dict	119754
merged	**249056**

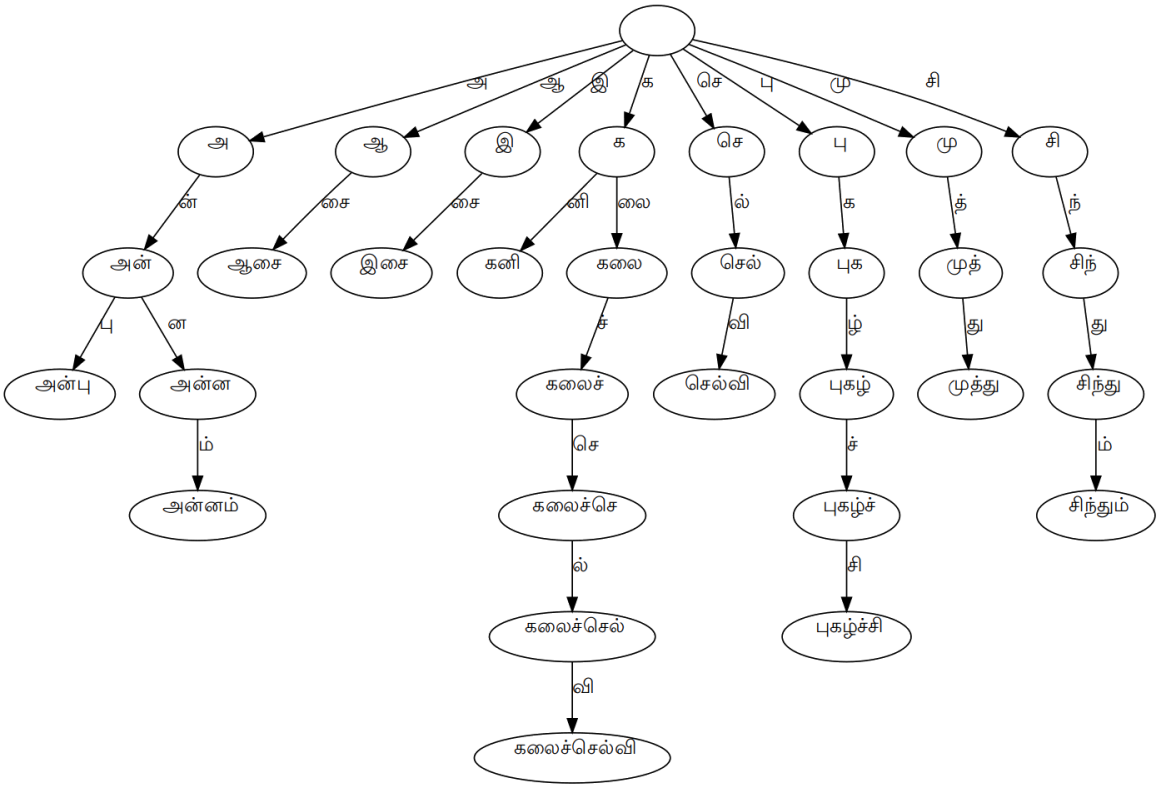


Fig-1: Trie built for the Tamil words: அன்பு , அன்னம் , ஆசை, இசை, கனி, கலைச்செல்வி, செல்வி, புகழ், புகழ்ச்சி, முத்து, சிந்து, சிந்தும். Notice that the செல்வி in கலைச்செல்வி is completely different from standalone செல்வி

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, *அன்பு , அன்னம் , ஆசை, இசை, கனி, கலைச்செல்வி, செல்வி, புகழ், புகழ்ச்சி, முத்து, சிந்து, சிந்தும்.*

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.

இச	*இசை*	செல்வ	*செல்வி*	முத்ை	*முத்து*
இசை	*இசை*	செல்வி	*செல்வி*	மு்து	*முத்து*
ைஇ	*இசை*	செல்ை	*செல்வி*	லச்ச	*கலைச்செல்வி*
கன	*கனி*	செ்வி	*செல்வி*	ைந்து	*சிந்து, சிந்தும்*
கனி	*கனி*	சை	*இசை*	ைகழ்	*புகழ்*
கலச்ச	*கலைச்செல்வி*	னி	*கனி*	ைகழ்ச	*புகழ்ச்சி*
கலைச	*கலைச்செல்வி*	பகழ்	*புகழ்*	ைத்து	*முத்து*
கலைச்	*கலைச்செல்வி*	பகழ்ச	*புகழ்ச்சி*	ெல்வி	*செல்வி*
கலைச்ச	*கலைச்செல்வி*	புகழ	*புகழ்*	அனனம்	*அன்னம்*
கலை்ச	*கலைச்செல்வி*	புகழச	*புகழ்ச்சி*	அனபு	*அன்பு*
கி	*கனி*	புகழ்	*புகழ், புகழ்ச்சி*	அன்னம	*அன்னம்*
கைச்ச	*கலைச்செல்வி*	புகழ்ச	*புகழ்ச்சி*	அன்னம்	*அன்னம்*
சந்து	*சிந்து, சிந்தும்*	புக்	*புகழ்*	அன்னம்	*அன்னம்*
சல்வி	*செல்வி*	புக்ச	*புகழ்ச்சி*	அன்ன்	*அன்னம்*
சிநது	*சிந்து, சிந்தும்*	புழ்	*புகழ்*	அன்ப	*அன்பு*
சிந்த	*சிந்து, சிந்தும்*	புழ்ச	*புகழ்ச்சி*	அன்பு	*அன்பு*
சிந்து	*சிந்து, சிந்தும்*	மத்து	*முத்து*	அன்ம்	*அன்னம்*
சிந்ை	*சிந்து, சிந்தும்*	முதது	*முத்து*	அன்ை	*அன்பு*
சிது	*சிந்து, சிந்தும்*	முத்த	*முத்து*	அ்னம்	*அன்னம்*
செலவி	*செல்வி*	முத்து	*முத்து*	அ்பு	*அன்பு*
ஆச	*ஆசை*	கலைச்ச	*கலைச்செல்வி*	ன்பு	*அன்பு*
ஆசை	*ஆசை*	சிந்து	*சிந்து, சிந்தும்*	புகழ்	*புகழ்*
ைஆ	*ஆசை*	செல்வி	*செல்வி*	புகழ்ச	*புகழ்ச்சி*
இசை	*இசை*	சை	*ஆசை*	முத்து	*முத்து*
கனி	*கனி*	ன்னம்	*அன்னம்*		

Fig-2: Sample edit lookup table built by symspell with edit distance = 2 for □□□□□, □□□□□□, □□□, □□□, □□□, □□□□□□□□□□□, □□□□□□, □□□□□, □□□□□□□□□, □□□□□□, □□□□□□, □□□□□□□□.

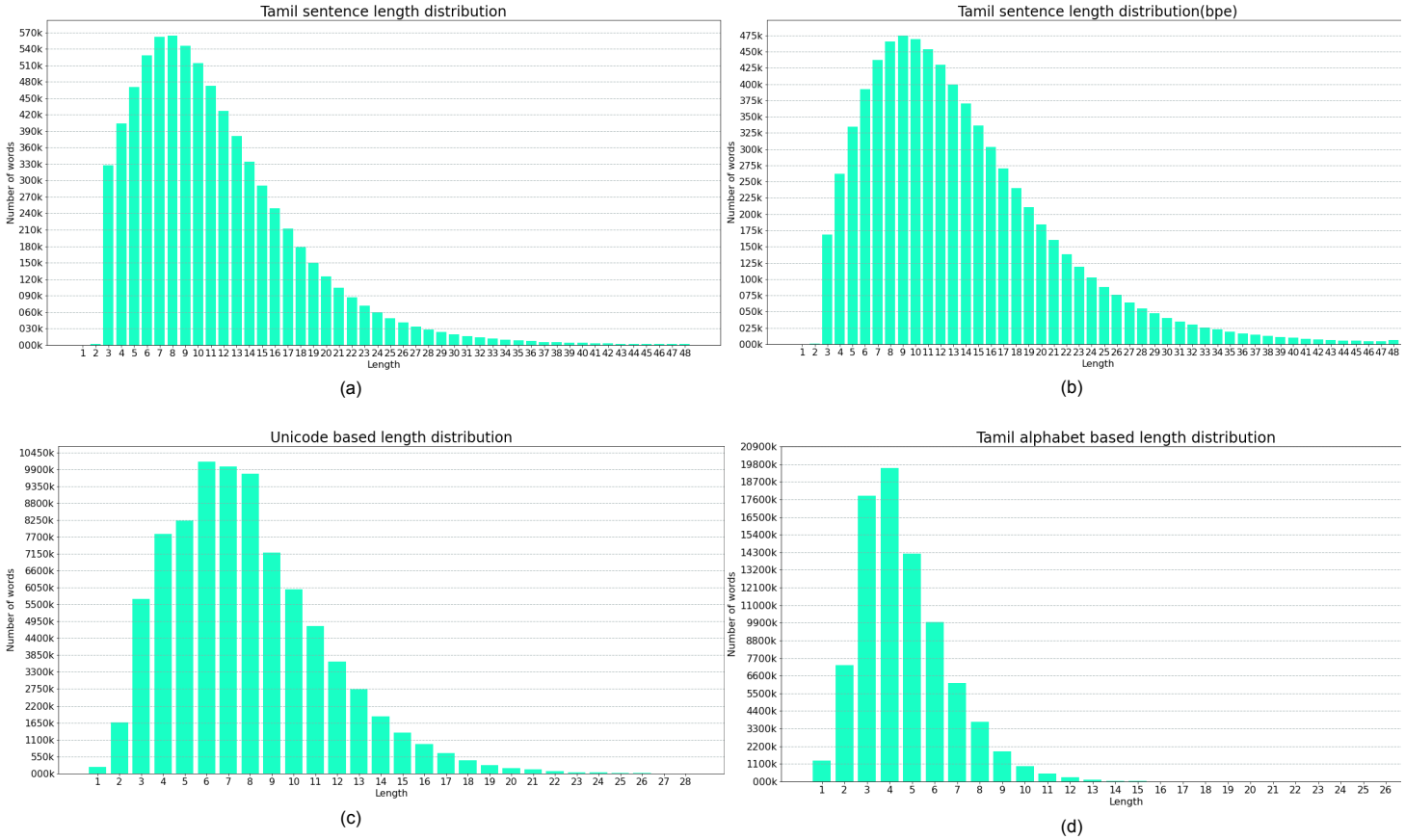


Fig-3: Distribution of sentence and word lengths in the corpus *tamiltext-7M.txt*. (a) The space delimited sentence length(number of words in a sentence) distribution. (b) BPE tokenized sentence length distribution. (c) word length distribution based on unicode encoding. (d) word length distribution based on natural Tamil alphabets.

```
-- Code Listing --
BuildCorruptedSamples(alphabets, dictionary, text, abberation_count):
  tokens = split_by_whitespace(text)
  for i in range(0, length(tokens)):
    if abberation_count:
      if random_float() > 0.5:
        abberation_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 l*earning_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. Symspell 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 spell-checkers 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

1. László Németh, Kevin Hendricks, *https://github.com/hunspell/hunspell*
2. Sepp Hochreiter and Jürgen Schmidhuber, *Long Short-Term memory*. Neural computation, 1997.
3. Yonghui Wu, et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*, 2016
4. Sowmya S Sundaram, *Morphological Processing for தமிழ் — The Unsupervised Way,* https://medium.com/analytics-vidhya/morphological-processing-for-தமிழ்-the-unsupervised-way-68afebc388c4
5. T. Shrinivasan, et al., *all_nouns.txt*, https://github.com/KaniyamFoundation/all_tamil_nouns
6. *Tamil Etymological Dictionary*, https://github.com/vanangamudi/tharavukkanam
7. Malaikannan Sankarasubbu, *Bloom-filter*, https://github.com/malaikannan/TamilSpellChecker
8. Wolf Garbe, *Python implementation of SymSpell algorith*m, https://github.com/mammothb/symspellpy
9. Selvakumar Murugan, *tamiltext-7M.txt*, https://github.com/vanangamudi/tharavukkanam/tree/master/tamil-etymological-dict
10. Ashish Vaswani, et al., *Attention Is All You Need*, https://arxiv.org/abs/1706.03762

பத்தொன்பதாவது தமிழ் இணைய மாநாடு, 2020
டிசம்பர் 11, 12 & 13.

A Survey on Neural Machine Translation for English-Tamil Language pair

B.Janarthanasarma, T.Uthayasanker
Department of Computer Science and Engineering, University of Moratuwa
{janarthanasarma.13,rtuthaya}@cse.mrt.ac.lk

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 , it's 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.

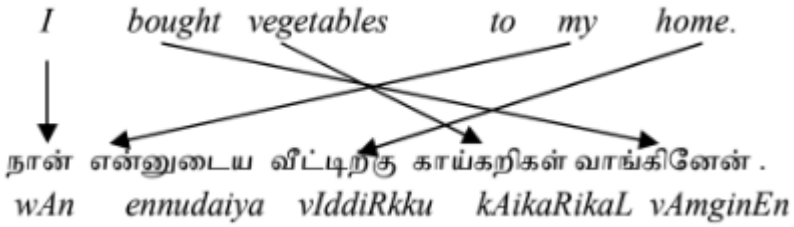


Fig 1. Word order difference between English and Tamil

Root	Tenses	Word Form
விளையாடு (vilayAdu)	Present+1S	விளையாடுகின்றேன் vilayAdu-kinR-En
	Present+3SN	விளையாடுகின்றது vilayAdu-kinR-athu
	Present+3PN	விளையாடுகின்றன vilayAd-kinR-ana
	Past+1S	விளையாடினேன் vilayAd-in-En
	Past+3SM	விளையாடினான் vilayAd-in-An
	Future+2S	விளையாடுவாய் vilayAdu-v-Ay
	Future+3SF	விளையாடுவாள் vilayAdu-v-AL

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(Turchi et al., 2008; Irvine and CallisonBurch, 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.

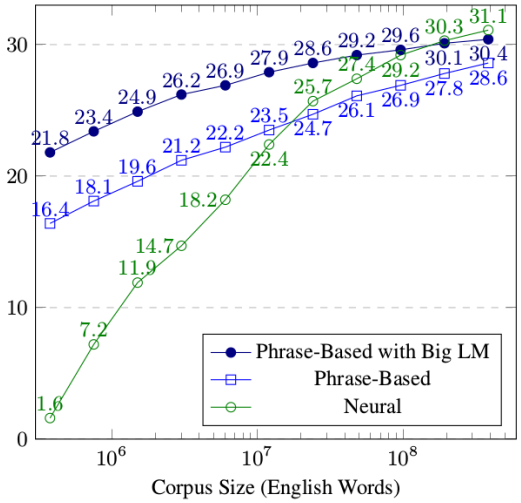


Figure 2: The quality of SMT and NMT in low-resource conditions as observed by Koehn and Rebecca Knowles.2017

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),WordPieces(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 Kunchukkutan 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

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, pages 3104–3112, Montreal, Quebec, Canada.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the International Conference on Learning Representations (ICLR).
[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In Advances in Neural Information Processing Systems 30, pages 5998–6008.
[4] Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In Proceedings of the First Workshop on Neural Machine Translation, pages 28–39, Vancouver.
[5] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-Based & Neural Unsupervised Machine Translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 5039–5049, Brussels, Belgium.
[6] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. ¨ Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," arXiv:1503.03535v2, 2015. [Online]. Available: http: //arxiv.org/abs/1503.03535
[7] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," arXiv:1511.06709v4, 2016. [Online]. Available: http://aclweb.org/anthology/P/P16/P16-1009.pdf
[8] Y. Cheng, W. Xu, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, "Semi-supervised learning for neural machine translation," Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, p. 19651974, 2016. [Online]. Available: https://www. aclweb.org/anthology/P16-1185
[9] J. Zhang and C. Zong, "Exploiting source-side monolingual data in neural machine translation," Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, p. 15351545, 2016. [Online]. Available: https://www.aclweb.org/anthology/D16-1160
[10] A. Currey, A. V. M. Barone, and K. Heafield, "Copied monolingual data improves low-resource neural machine translation," Proceedings of the Conference on Machine Translation (WMT), Volume 1, p. 148156, 2017. [Online]. Available: https://www.aclweb.org/anthology/W17-4715
[11] T. Domhan and F. Hieber, "Using target-side monolingual data for neural machine translation through multi-task learning," Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, p. 15001505, 2017. [Online]. Available: https://www. aclweb.org/anthology/D17-1158
[12] F. Stahlberg, J. Cross, and V. Stoyanov, "Simple fusion: Return of the language model," arXiv:1809.00125v2, 2019. [Online]. Available: https://arxiv.org/abs/1809. 00125
[13] P. Ramachandran, P. J. Liu, and Q. V. Le, "Unsupervised pretraining for sequence to sequence learning," arXiv:1611.02683v2, 2018. [Online]. Available: https://arxiv.org/abs/1611.02683
[14] I. Skorokhodov, A. Rykachevskiy, D. Emelyanenko, S. Slotin, and A. Ponkratov, "Semi-supervised neural machine translation with language models," Proceedings of AMTA 2018 Workshop: LoResMT 2018, pp. 37–44, 2018. [Online]. Available: http://sereja.me/f/loresmt.pdf
[15] Tom Kocmi and Ondˇrej Bojar. 2018. Trivial Transfer Learning for Low-Resource Neural Machine Translation. In Proceedings of the Third Conference on Machine Translation, pages 244–252, Belgium, Brussels.
[16] Toan Q. Nguyen and David Chiang. 2017. Transfer Learning across Low-Resource, Related Languages for Neural Machine Translation. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 296–301, Taipei, Taiwan.
[17] Graham Neubig and Junjie Hu. 2018. Rapid Adaptation of Neural Machine Translation to New Languages. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 875–880, Brussels, Belgium.

[18] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer Learning for Low-Resource Neural Machine Translation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1568–1575, Austin, Texas.
[19] Yun Chen, Yang Liu, Yong Cheng, and Victor O.K. Li. 2017. A Teacher-Student Framework for ZeroResource Neural Machine Translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1925–1935, Vancouver, Canada.
[20] Yun Chen, Yang Liu, and Victor O. K. Li. 2018. Zero-Resource Neural Machine Translation with Multi-Agent Communication Game. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. AAAI Press, 5086–5093.
[21] Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. 2017. Joint Training for Pivot-based Neural Machine Translation. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. Melbourne, 3974–3980. https: //doi.org/10.24963/ijcai.2017/555
[22] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-Task Learning for Multiple Language Translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, Beijing, China, 1723–1732. https://doi.org/10.3115/v1/P15-1166
[23] Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, San Diego, California, 866–875. https://doi.org/10.18653/v1/N16-1101
[24]Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In Proceedings of the First Workshop on Neural Machine Translation, pages 28–39.
[25][Anand Kumar et al.2014]Kumar, M Anand, V Dhanalakshmi, K P Soman, and V Sharmila Devi. 2014. Improving the Performance of English-Tamil Statistical Machine Translation System using Source-Side Pre-Processing. arXiv preprint arXiv:1409.8581.
[26] Anand Kumar, M., Dhanalakshmi, V., Rekha, R. U., Soman, K. P., & Rajendran, S. (2010). A Novel Data Driven Algorithm for Tamil Morphological Generator. International Journal of Computer Applications, 52-56.
[27]Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 211–221. Association for Computational Linguistics

லஸ்ஸி - உங்கள் தாய்மொழியில் நிரலாக்கம்

ம. ஜூலீஎன் (Julien Malard)[க], ஹா. யோல் (Joel Harms)[க], ஆ. யான் (Jan Adamowski)[க], மெ-கீ ஊகோ (Hugo Melgar-Quiñonez)[உ]

[க] உயிர் வள பொறியியல் துறை, மக்கில் பல்கலைக்கழகம், கனடா

[உ] உலக உணவு பாதுகாப்பு நிலையம், மக்கில் பல்கலைக்கழகம், கனடா

தொடர்பு மின்னஞ்சல் முகவரிகள் - julien.malard@mail.mcgill.ca; joel.harms@mail.mcgill.ca; jan.adamowski@mcgill.ca; hugo.melgar-quinonez@mcgill.ca

சுருக்கம்

உலகத்தில் பிரபலமான நிரல்மொழிகள் பெரும்பான்மையாக ஆங்கில மூலமாக எழுதப்பட்டுள்ளன. வேறு இயற்கை மொழிகளில் நிரலாக்கம் செய்யும் முயற்சிகள் பல்வேறு இருந்தாலும், தொழில்முறை சூழல்நிலையில் அவற்றின் பயன்பாடு குறைவு. இந்த கட்டுரையில் **லஸ்ஸி** (https://லஸ்ஸி.இந்தியா) என்று சட்டகம் மற்றும் மென்பொருள் கருவி முன்வைக்கப்படும். லஸ்ஸியால் எந்த நிரல்மொழி வேண்டுமானாலும், உலகத்தில் உள்ள எல்லா இயற்கை மொழிகளிலும் எழுதுதல் சாத்தியமாகிறது. இந்த கருவியால் நிரலாக்க புலத்தில் வாய்ப்பு சமத்துவம் அதிகரிக்கும் என்று நம்புகிறோம்.

க. முன்னுரை

சமீபத்திய காலங்களில் நிரலாக்க புலத்தில் இயற்கை மொழி பன்முகத்தன்மையின் முக்கியத்துவத்தின் அங்கீகாரம் அதிகரிக்கிறது. தாய்மொழியில் கணினி அறிவியல் கற்றுகொண்டு குழந்தைகளுக்கு கல்வியில் நன்மை உண்டு என்று ஏற்கனவே நிரூபிக்கப்பட்டுள்ளது (Dasgupta மற்றும் Hill உசூஎ). ஆயினும் கணினியியல் புலத்தில் பெரும்பான்மை முயற்சிகள் பயனர் இடைமுக மொழியாக்கத்துக்காக மட்டுமே நாடுகின்றன, அதே நேரத்தில், நிரல்மொழிகள் பெரும்பான்மையில் ஆங்கிலத்தில் மட்டும் எழுதப்பட்டுள்ளன (McCulloch உசூஎக). இந்த சூல்நிலையில், உலகத்தின் எல்லா மனிதர்களால் மென்பொருட்களை வாங்கி உபயோக முடியும் என்றாலும், ஆங்கிலம் தெரிந்தவர்களுக்கு மட்டுமே நிரலாக்கம் கற்றுகொண்டு மென்பொருளை எழுதி விற்கும் வாய்ப்பு கிடைக்கும் (Smith IV உசூஎரு). இந்த சமத்துவமின்மையை குறைப்பதற்காக பல்வேறு நிரலாக்கம் சர்வதேசமயமாக்கல் முயற்சிகள் தற்காலத்தில் நடந்து கொண்டு இருக்கின்றன.

ஆங்கில சாராத நிரலாக்கத்துக்காக இரண்டு பொதுவான உத்திகள் உள்ளன. முதலாவது, புதிய சுயாதீனமான நிரல்மொழிகள் உருவாக்கலாம், அதை தவிர ஏற்கனவே உபயோகத்தில் உள்ள ஆங்கில நிரல்மொழிகளின் மொழிபெயர்ப்பு செய்யலாம்.

பல்வேறு ஆங்கில சாராத நிரல்மொழிகள் உருவாக்கப்பட்டுள்ளன. உதாரணத்துக்காக, சமீபத்திய காலங்களில் எழில் (தமிழ்) (Muthiah மற்றும் Annamalai உசூஎச), YorLang (யொரூபா) (Anuoluwapo உசூஎக), なでしこ (சப்பானிய) (なでしこ உசூஎஅ), قلب (அரபு) (ناصر உசூஎஙு), Swahili (சுவாகிலி) (Kiano மற்றும் Wendo உசூஎஅ), Egua (போர்த்துக்கேய) (Egua உசூஎஅ), Linotte (ஃபிரெஞ்சு) (Linotte உசூஎஅ), 文言 (சீன் மொழி) (文言 உசூஎஅ), کاتی (பார்சீகம்) (کاتی உசூஎஅ), कलाम (இந்தி, மராத்தி) (कश्म உசூஎஅ), ஆகிய நிரல்மொழிகள் உபயோகத்தில் வந்து விட்டன. இவற்றை தவிர Citrine (சீற்றின்) என்று நிரல்மொழி சிறப்பு கவனம் தகுதியானது, ஏனென்றால் அது ஒரே மொழியில் இல்லை, ஆனால் நேரடியாக நூற்றி எட்டு மொழிகளில்

வெளியிடப்பட்டுள்ளது (de Mooij மற்றும் Jilani உஆஊ). ஆங்கில சாராத நிரல்மொழிகளின் முழு பட்டியலுக்காக விக்கிப்பீடியாக்கு செல்லவும் (« தமிழ் விக்கிப்பீடியா » உஆஐஙூ).

ஏற்கனவே உபயோகத்தில் உள்ள நிரல்மொழிகளின் மொழியாக்கத்தின் உத்தி பக்கத்திலும் பல்வேறு உதாரணங்கள் கிடைக்கும். இந்த உத்தி ஏற்கனவே ரியாக்ட (React உஆஉ) அல்லது காஃபிக்கிறிட்டு (Ashkenas உஆஉ) என்பது தொழில்நுட்பங்களால் ஊக்குவிக்கப்பட்டுள்ளது. இருந்தாலும், இவற்றின் விஷயத்தில் ஆங்கில மூலமான ஒரு நிரல்மொழி வேறொன்று ஆங்கில மூலமான நிரல்மொழிக்கு மொழிபெயர்ப்படுகிறது. வேறொன்று இயற்கை மொழிக்கு மொழிபெயர்க்கப்படும் மொழிகளின் உதாரணங்களில், ঢ ঞ্ঝ (வங்காளத்தில் யாவாக்கிறிட்டு) (ক্সাব மற்றும் பலர் உஆஐச), 周蟒 (சீன் மொழியில் பைத்தான்) (蓋索林 உஆஐஙூ), பைத்தம் (தமிழ் பைத்தான்) (கணேஷ் குமார் உஆஙூ), மற்றும் Babylscript (பல்வேறு மொழிகளில் யாவாக்கிறிட்டு) (Iu உஆஐக) என்று நிரலாக்க மொழிகள் வரும். CodeInternational என்று இன்னொரு கருவி (Piech மற்றும் Abu-El-Haija உஆஐகூ) குறியீட்டில் உள்ள இனங்காட்டிகளை மொழிபெயர்க்கும், ஆனால் நிரல்மொழியின் சிறப்பு சொற்களை மொழிபெயர்க்காது.

புது நிரல்மொழி உருவாக்கத்துடன் ஒப்பிடுகையில் இந்த உத்திக்கு சில பலங்கள் உள்ளன. முதலில், ஏற்கனவே உபயோகத்தில் உள்ள மொழிகளுக்கு பல்வேறு செயல்பாடுகளுக்காக எழுதப்பட்ட, வெளிப்புற பொதிகள் கிடைக்கும். தொகுப்பியின் செயல்படுத்தலை பொறுத்து, மொழிபெயர்ப்பட்ட நிரல் மூலமொழியில் எழுதப்பட்ட நிரல் பொதிகளுடனும் இணக்கமானதாக இருக்கும். இந்த வழியால் ஏற்கனவே செய்யப்பட்ட நிரலாக்க முயற்சிகளின் நகல் தவிர்க்கப்படும். இரண்டாவது, உண்மையான, உலகில் புதிதாக உருவாக்கப்பட்ட மொழிகளை பயன்படுத்துவதற்கான வாய்ப்புகள் குறைவு. இதற்கு பல புது நிரல்மொழிகள் கல்விமொழியால் தானே வழங்கப்படுகின்றன (Linotte உஆஉ; Muthiah உஆஐஙூ). இணையத்தள பயன்பாடு வளர்ச்சிக்கு யாவாக்கிறிட்டு தேவை, அதே மாதிரி, பைத்தான் இல்லாத அறிவியல் நிரலாக்க வாய்ப்புகளும் குறைவு. நிரல்மொழி மொழியாக்கத்தால் இந்த முக்கியமான நிரல்மொழிகளை பல்வேறு இயற்கை மொழிகள் மூலமாக பயன்ப்படுத்தலாம். இறுதியில், நிரல்மொழி மொழியாக்க உத்தி புதிய நிரல்மொழி உருவாக்க உத்தியுடன் இணைக்கக்கூடியது என்று நினைவில் கொள்ளவும். அதாவது, ஆங்கில் சாராத வேறு ஒரு மொழியில் உருவாக்கப்பட்ட நிரல்மொழியையும் வெவ்வெறு இயற்கை மொழிகளில் மொழிபெயர்க்கலாம்.

ஆயினும், இன்று வரை நிரல்மொழியின் மொழியாக்கத்துக்காக ஒரு பொதுவாதி சட்டகம் கிடையாது. இந்த கட்டுரையில் நாங்கள் **லஸ்ஸி** என்று உலகத்தில் முதல் பொதுவாதி நிரல்மொழி மொழியாக்க சட்டகமும் மென்பொருள் கருவியும் முன்வைக்கிறோம். இந்த மென்பொருளால் உலகத்தில் பேசப்பட்ட ஒவ்வொரு இயற்கை மொழியிலும் எந்த நிரல்மொழி வேண்டுமானாலும் எழுதலாம்.

உ. லஸ்ஸி சட்டகத்துக்கு முதல் அறிமுகம்

நிரல்மொழி மொழியாக்கத்துக்காக மூன்று படியுள்ள ஒரு முறை முன்வைக்கிறோம்:

க. மூலக் குறியீடு பகுப்பாய்வியால் **குறியீடு மர வரைபடம்** உருவாக்கம் (« தமிழ் விக்கிப்பீடியா » உஆஐச)

உ. மரத்தின் **இனங்காட்டிகள்** மற்றும் **எண்ணுரு முறைமை** மாற்றம்

ங. மரம் மூலம் மொழிபெயர்க்கப்பட **குறியீடு உருவாக்கம்** (சிறப்பு சொற்கள், வாக்கிய அமைப்பு, நிறுத்தற்குறிகள்)

இந்த மூன்று படிகள் லஸ்ஸி என்ற மென்பொருளால் செயல்படுத்தப்பட்டுள்ளன. லஸ்ஸியின் முக்கிய இணைதளப் பக்கம் இந்த முகவரியில் கிடைக்கும் - https://லஸ்ஸி.இந்தியா. மூல நிரல் கிட்ஹபில் திறந்த மூலமாக கிடைக்கும் - https://github.com/lassi-samaaj.

தற்காலத்தில் லஸ்ஸியில் பைத்தானுக்காக பதினொரு மொழிகள் கிடைக்கும், ஜெஸானுக்காக ஐந்து மொழிகள் கிடைக்கும். யாவாக்கிறிட்டையும் எழிலையும் சேர்ப்பதற்கு

வேலை நடத்து கொண்டு இருக்கிறது. லஸ்ஸி சட்டகத்தின் செயல்படுத்தல் பைத்தான் மொழியில் எழுதப்பட்டுள்ளது.

உ.க குறியீடு மரம் உருவாக்கம்

முதல் படியில், லார்க் (Shinan உசூஉய) என்று இலக்கண பகுப்பாய்வியால் உள் குறியீட்டின் மரம் உருவாக்கப்படும். இந்த மரத்தில் உள் குறியீட்டின் ஒவ்வொரு வெளிப்பாடும் ஒரு கிளையால் குறிப்பிடப்படுகிறது.

உ.உ இனங்காட்டிகள் மற்றும் எண்ணுரு முறைமை மாற்றம்

இரண்டாம் படி விருப்பமானது. இந்த படியில், லஸ்ஸி குறியீடு மரத்தில் ஒவ்வொரு கிளைக்கு வருகை செய்து, எண் அல்லது இனங்காட்டிக்கு தொடர்புடையான இலைக்கு தேவைப்பட்ட மாற்றம் செய்யும். மூலமொழியும் வேண்டிய மொழியும் ஒரே எண்ணுரு முறைமை பயன்படுத்தினால், எண்ணுரு மாற்றம் தேவையாக இருக்காது. அதே மாதிரி, கணினியில் குறியீடு செயல்படுத்தலுக்கான இனங்காட்டி மொழியாக்கம் தேவை இல்லை.

எண்ணுரு முறைமை மொழியாக்கம் எண்ணிக்கை (ஜூலீஎன் உசூஉய) என்று நிரல்மொழியால் நிர்வகிக்கப்பட்டுள்ளது. இயல்பாக உள்ளமைப்புடன் இயற்கை மொழி தொடர்புடைய எண்ணுரு முறைமை பயன்படும், அதாவது, லஸ்ஸி தமிழ் பைத்தானில் தமிழ் எண்களை எதிர்பார்க்கும், அதே மாதிரி, இந்தி பைத்தானில் தேவனாகரி எண்களை எதிர்பார்க்கும். ஆயினும், பயனாளரால் இந்த நடத்தை சொந்த விருப்பத்துக்கு மாறலாம். ஏதோ ஒரு வினோதமான காரணத்துக்காக ஒருவர் தமிழ் பைத்தானில் குஜராத்தி எண்ணுரு பயன்படுத்த விரும்பினால், லஸ்ஸிக்கு புரியும்.

உ.ங குறியீடு உருவாக்கம்

மூன்றாம் படியில் லார்க் புனரமைப்பு செயல்பாட்டால் வேண்டிய மொழியின் இலக்கணத்துடன் குறியீடு மரத்தால் வேண்டிய குறியீடு உரை உருவாக்கப்படும். இந்த படியில் இரண்டு வகையான மாற்றங்கள் மேற்கொள்ளப்படும்: சிறப்பு சொற்கள் மற்றும் இலக்கண அமைப்பு.

உதாரணத்துக்காக, ஆங்கில பைத்தானில்

```
return ஆ
```

தமிழில் ஆகும்

```
பின்கொடு ஆ
```

இடாய்ச்சு மொழியில் *பின்கொடு* என்று சிறப்பு சொல்லின் இடையில் இரண்டு சிறப்பு சொற்கள் தேவையானது:

```
gibt ஆ zurück
```

இந்த உதாரணத்தில், சிறப்பு சொற்கள் இரண்டுமே லஸ்ஸியால் மாற்றப்பட்டன. தமிழில் உதாரணத்துக்காக, ஆங்கில பைத்தானின்

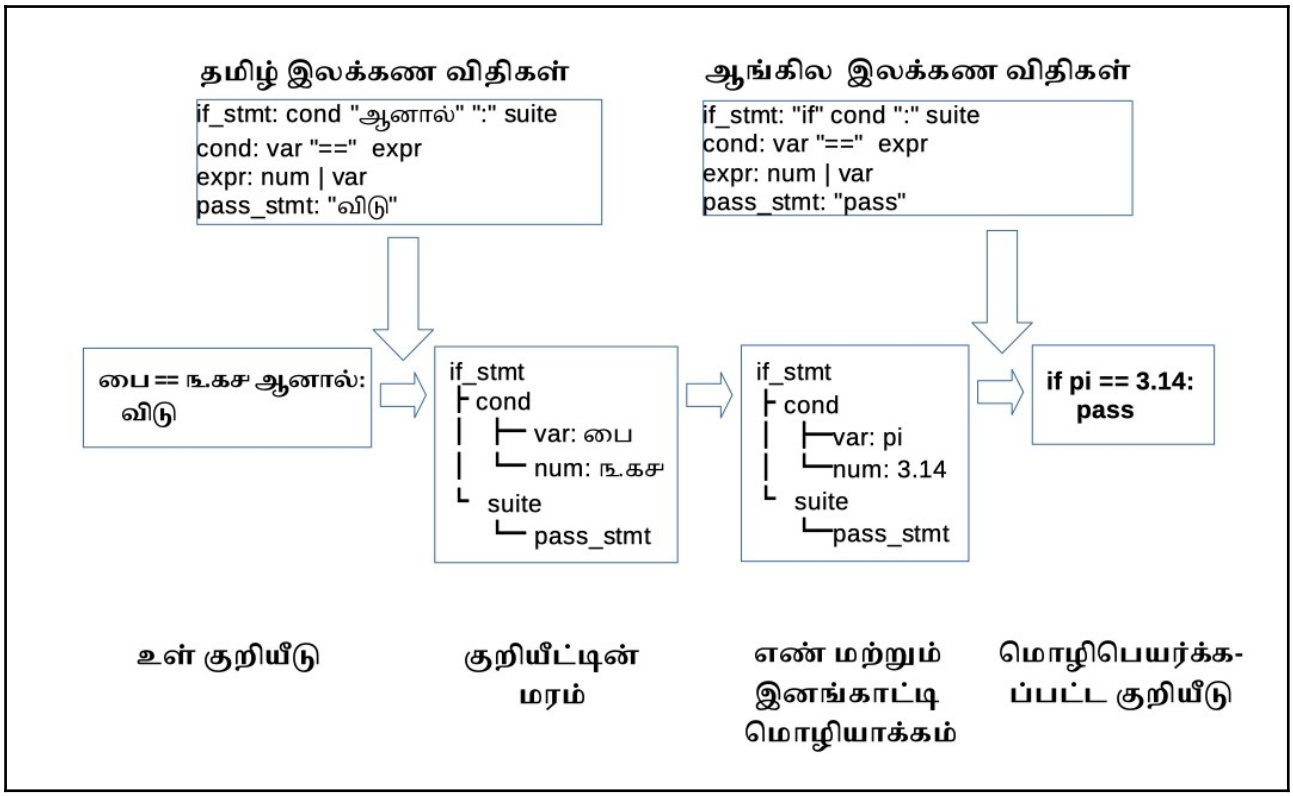
```
if விலை > ௦:
        pass
```

தமிழில் ஆகும்

```
விலை > ய ஆனால்:
        விடு
```

இந்த செயல்பாடுகள் லார்கின் சக்திவாய்ந்த பகுப்பாய்வி வழிமுறையால் (லால்ர்/lalr மற்றும் எர்லீ/earley) சாத்தியமானது. பெரும்பாலான வேறு நிரல்மொழியாக்க கருவிகள் குறியீடு மரத்தை உருவாக்காமல் வெறுமனே சிறப்பு சொற்களை சொந்த இடத்திலே நேரடியாக மொழிபெயர்கின்றன. லஸ்ஸி அதற்கு பதிலாக முழு வெளிப்பாட்டை புரிந்து இயற்கையான முறையால் மொழியாக்கம் கொடுக்கிறது.

கீழ்கண்ட உருப்படம் க இல் லஸ்ஸி சட்டகத்தின் இயங்கு முறை விளக்கப்பட்டுள்ளது. உருப்படம் உ இல் லஸ்ஸியால் படிக்க கூடிய தமிழ் பைத்தானும் இந்தி பைத்தானும் விளக்கப்பட்டுள்ளன.



உருப்படம் க: உதாரண தமிழ் பைத்தான் குறியீட்டுடன் லஸ்ஸி சட்டகத்தின் இயங்கும் முறை விளக்கம்.

```
தொகுப்பு வட்டம்_தொகுப்பு(பொருள்):
  பை = ங.கசகருகூஉசுருங

  நிரல்பாகம் __துவக்கம்__(தன், ஆரம்):
    தன்.ஆரம் = ஆரம்

  நிரல்பாகம் சுற்றளவு(தன்):
    பின்கொடு உ * தன்.பை * தன்.ஆரம்

  நிரல்பாகம் பரப்பளவு ( தன் ) :
    பின்கொடு தன்.பை * தன்.ஆரம் ** உ

ஆரங்கள் = சரகம்(ரு)
வட்டங்கள் = [
  வட்டம்_தொகுப்பு(ஆரம்=ஆ)
  ஒவ்வொன்றாக ஆ ஆரங்கள் இல்
]

ஒவ்வொன்றாக வ வட்டங்கள் இல்:
  பதிப்பி(வ.சுற்றளவு(), வ.பரப்பளவு())
```

```
वर्ग वृत्त_वर्ग(वस्तु):
  पाई = ३.१४१५९२६५३

  फलन __प्रारंभ__(स्वयं, त्रिज्या):
    स्वयं.त्रिज्या = त्रिज्या

  फलन परिधि(स्वयं):
    लौटा २ * स्वयं.पाई * स्वयं.त्रिज्या

  फलन क्षेत्रफल(स्वयं):
    लौटा स्वयं.पाई * स्वयं.त्रिज्या ** २

त्रिज्याएं = श्रृंखला(५)
वृत्तएं = [वृत्त_वर्ग(त्रिज्या=त्रि) हर त्रि त्रिज्याएं से]

हर वृ वृत्तएं से:
  चपाना(वृ.परिधि(), वृ.क्षेत्रफल())
```

உருப்படம் உ : லஸ்ஸி தமிழ் பைத்தானில் எழுதப்பட்ட ஒரு உதாரண நிரல். இடது - தமிழ் பைத்தான். வலது - அதே குறியீடு, இந்தி பைத்தானில். இரண்டுமே இணையான, செல்லுபடியாகும் லஸ்ஸி பைத்தான் குறியீடுகள் ஆகும்.

உ.ச லஸ்ஸி மொழி விவரக்குறிப்பு

ஒவ்வொரு நிரலாக்க மொழிக்காகவும், லஸ்ஸிக்கு அந்த மொழியின் இலக்கண விவரக்குறிப்பு தேவையானது. இந்த விவரக்குறிப்பு லார்க் (Shinan உசூஉ) என்று வடிவத்தில் கொடுக்கவும்.

லஸ்ஸி இந்த இலக்கண சட்டங்களை படித்து மொழிபெயர்ப்புக்கு ஒரு ஜெஸான் வடிவ கோப்பு உருவாக்கும். இந்தக் கோப்பை திருத்தி மொழிபெயர்ப்புகளை சேர்க்கலாம்.

உதாரணத்துக்காக, மூல ஆங்கிலத்தில் இந்த விதி

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

தமிழில் ஆகும்

```
while_stmt : test "வரை" ":" suite [ "ஏதேனில்" ":" suite ]
```

இந்த மொழிபெயர்க்கப்பட்ட விதிகளுடன் லஸ்ஸியால் குறியீட்டை மொழிபெயர்க்க முடியும்.

ங. முடிவுரை

நிரலாக்க புலத்தில் சேர்க்கை மற்றும் வாய்ப்பு சமத்துவத்துக்காக இயற்கை மொழி ரீதியாக சுயாதீனமான நிரலாக்க கருவிகள் தேவை. இந்த கட்டுரையில் லஸ்ஸி என்று உலகத்தில் முதல் இயற்கை மற்றும் நிரலாக்க மொழி ரீதியாக பொதுவாதி குறியீடு மொழிப்பெயர்ப்பாளர் விளக்கப்பட்டுள்ளது. ஏற்கனவே உபயோகத்தில் உள்ள கருவிகளுடன் ஒப்பிடுகையில் லஸ்ஸி சில புது நன்மைகளை வழங்குகிறது. முதல் இடத்தில், பிரபலமான ஆங்கில

நிரல்மொழிகளுடனும் அவற்றின் வெளிப்புற பொதிகளுடனும் லஸ்ஸி நூறு சதவீதமாக இணைக்கமானது. இந்த வழியால் லஸ்ஸியின் தத்தெடுப்பிலும் பகிர்விலும் வேறு புது நிரல்மொழிகளுக்கு வந்துள்ள பொருந்தக்கூடிய தன்மையின் சிரமங்கள் வராது (Smith IV உஆஇ).

லஸ்ஸியில் உபயோகிக்கப்பட்ட நிரல்மொழி பகுப்பாய்வி வழிமுறைகளால் சிறப்பு சொற்கள் மட்டும் அல்ல, வெளிப்பாட்டின் முழு இலக்கணத்தை மொழிபெயர்க்கலாம். இந்த தன்மை, ஏற்கனவே உபயோகத்தில் உள்ள நிரல்மொழி மொழியாக்க கருவிகளில் (Iu உஆஇ; [illegible] et al. உஆஅ) கிடையாது. அது தவிர, லஸ்ஸியின் ஒருங்கிணைந்த மொழியாக்க முறையால் புது கணினி அல்லது இயற்கை மொழி சேர்தல் சுலபமான விஷயம் ஆகும்.

இறுதியில், லஸ்ஸி ஒரு கல்வி நிரலாக்க மொழி அல்ல. பிரபலமான நிரல்மொழிகளுடன் லஸ்ஸியின் பொருந்தக்கூடிய தன்மையால் எல்லோராலும் சொந்த தாய்மொழியில் குறியீடு உருவாக்கி, அதை லஸ்ஸி மூலமாக இன்னொரு மொழிக்கு மொழிபெயர்த்து கூட்டுப்பணியாளர்களுக்கோ, முதலாளிக்கோ, வாடிக்கையாளருக்கோ அனுப்பலாம். பெரும்பான்மை மாணவர்களுக்கு நிரலாக்கம் கற்றலின் முதல் நன்மை வேலைவாய்ப்பு ஆகிறதால், இந்த சாத்தியம் லஸ்ஸியின் முக்கியமான நன்மைகளில் ஒன்று ஆகும்.

எதிர்காலத்தில், லஸ்ஸிக்கு அதிகம் நிரல்மொழிகளும் இயற்கை மொழிகளும் சேர்க்கப்படும். யாவாக்கிறிட்டிலும் ஒரு செயல்படுத்தல் உருவாக்கும் திட்டமும் உள்ளது. லஸ்ஸியின் வளர்ச்சிக்கு பங்களிப்புகளை வரவேற்கிறோம்.

சு. மேள்கோள்

Anuoluwapo, Karounwi. உஆஎக. *YorLang*. https://anoniscoding.github.io/yorlang.

Ashkenas, Jeremy. உஆஉஎ. *CoffeeScript*. https://coffeescript.org/.

Dasgupta, Sayamindu மற்றும் Benjamin Mako Hill. உஆஎள. « Learning to Code in Localized Programming Languages ». *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, நயஙந–நயகக. L@S'17. நியூயார்க், நியூயார்க், அமெரிக்க ஐக்கிய நாடுகள்: Association for Computing Machinery. https://doi.org/10.1145/3051457.3051464.

Egua. உஆஉஎ. https://egua.tech/.

Iu, Ming-Yee. உஆஎக. « Babylscript: multilingual javascript ». Dans *OOPSLA '11: Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, ஈகூஎள–ஈகூஎஅ. https://doi.org/10.1145/2048147.2048204.

Kiano, Malcolm மற்றும் Patrick Wendo. உஆஉஎ. *Swahili*. https://github.com/malcolmkiano/swahili.

Linotte. உஆஉஎ. langagelinotte.free.fr/.

McCulloch, Gretchen. உஆஎக. « Coding Is for Everyone—as Long as You Speak English ». *Wired*, 4 août உஆஎக. https://www.wired.com/story/coding-is-for-everyoneas-long-as-you-speak-english/.

Mooij, Gabor de மற்றும் Aavesh Jilani. உஆகக. *Citrine*. http://citrine-lang.org/.

Muthu Annamalai. உஆஎந. « Invitation to Ezhil: A Tamil Programming Language for Early Computer-Science Education ». உலகத் தமிழ்த் தகவல் தொழில்நுட்ப மன்றம் (உத்தமம்). மலேசிய. https://arxiv.org/abs/1308.1733.

Muthu Annamalai. உஆஎச. « Learning Ezhil Language via Web ». உலகத் தமிழ்த் தகவல் தொழில்நுட்ப மன்றம் (உத்தமம்). புதுச்சேரி. https://www.infitt.org/ti2014/papers/062_MAnnamalai_learning_ezhil_final.pdf.

Piech, Chris மற்றும் Sami Abu-El-Haija. உதயிக. « Human Languages in Source Code: Auto-Translation for Localized Instruction ». *arXiv:1909.04556 [cs]*. http://arxiv.org/abs/1909.04556.

React. உதஉய. Facebook. https://ta.reactjs.org/.

Shinan, Erez. உதஉய. *Lark*. https://github.com/lark-parser.

Smith IV, Jack. உதயிரு. « This Arabic Programming Language Shows How Computers Revolve Around the Western World ». Mic. உயஅ கார்த்திகை உதயிரு. https://www.mic.com/articles/130331/this-arabic-programming-language-shows-how-computers-revolve-around-the-western-world.

كاتى. உதஉய. https://www.scanf.ir/?page=kati.

ناصر, رمزي. உதயிஈ. قلب. https://github.com/nasser/---.

कदमस्वानंद. உதஉய. कलाम. https://www.kalaam.io.

সৈয়দ তানভীর জিসান, আরমান কামাল, মো. নুরুউদ্দিন মনসুর, மற்றும் মুরসালিন কবির. உதயிச. চা স্ক্রিপ্ট. https://sjishan.github.io/chascript/index.html.

« ஆங்கிலம் சாராத நிரலாக்க மொழிகள் ». உதயிஈ. தமிழ் விக்கிப்பீடியா. https://ta.wikipedia.org/w/index.php?title=ஆங்கிலம்_சாராத_நிரலாக்க_மொழிகள்

ஆர் கனேஷ் குமார். உதஈ. « Pytham: Python Pre-Processor Utility ». ாச-ாஅ. சென்னை. https://tamilnation.org/digital/tamilnet03/16_rganesh.pdf.

« இலக்கணப் பாகுபடுத்தி ». உதயிச. தமிழ் விக்கிப்பீடியா. https://ta.wikipedia.org/wiki/இலக்கணப்_பாகுபடுத்தி

ம. ஜூலீஎன். உதஉய. எண்ணிக்கை. https://github.com/julienmalard/ennikkai.

なでしこ. உதஉய. https://nadesi.com/doc3/index.php?なでしこの基本.

文言. உதஉய. https://github.com/wenyan-lang/wenyan.

蓋索林. உதயிஈ. 周蟒.

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 only 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. UBM 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. Disentangled 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. 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

[1] V. Z. Këpuska and H. A. Elharati, 'Robust Speech Recognition System Using Conventional and Hybrid Features of MFCC, LPCC, PLP, RASTA-PLP and Hidden Markov Model Classifier in Noisy Conditions,' J. Comput. Commun., vol. 03, no. 06, pp. 1-9, 2015, doi: 10.4236/jcc.2015.36001.

[2] S. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised speaker adaptation based on the cosine similarity for text-independent speaker verification," Odyssey 2010 Speak. Lang. Recognit. Work., no. May 2014, pp. 76–82, 2010.

[3] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc., vol. 2018-April, pp. 5329–5333, 2018, doi: 10.1109/ICASSP.2018.8461375.

[4] Y. Shi, Q. Huang and T. Hain, "H-Vectors: Utterance-Level Speaker Embedding Using a Hierarchical Attention Model," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 7579-7583, doi: 10.1109/ICASSP40776.2020.9054448.

[5] I. L.-M. Andrew Senior, "IMPROVING DNN SPEAKER INDEPENDENCE WITH I -VECTOR INPUTS," 2014 IEEE Int. Conf. Acoust. Speech Signal Process., pp. 225–229, 2014.

[6] C. Yu, A. Ogawa, M. Delcroix, T. Yoshioka, T. Nakatani, and J. H. L. Hansen, "Robust i-vector extraction for neural network adaptation in noisy environment," Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH, vol. 2015-January, pp. 2854–2857, 2015.

[7] S. Garimella, A. Mandal, N. Strom, B. Hoffmeister, S. Matsoukas, and S. H. K. Parthasarathi, "Robust i-vector based adaptation of DNN acoustic model for speech recognition," Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH, vol. 2015-Janua, pp. 2877–2881, 2015.

[8] Y. Miao, H. Zhang, and F. Metze, "Towards speaker adaptive training of deep neural network acoustic models," Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH, no. September, pp. 2189–2193, 2014.

[9] X. Li and X. Wu, "Modeling speaker variability using long short-term memory networks for speech recognition," in Interspeech, 2015.

[10] Xiaodong Cui, Vaibhava Goel, and George Saon, "Embedding-Based Speaker Adaptive Training of Deep Neural Networks," in Interspeech 2017. 122-126. 10.21437/Interspeech.2017-460.

[11] Xiaorui Wang Yuanyuan zhao, Jie Li and Yan Li, "The speechtransformer for large-scale mandarin chinese speech recognition," in 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019.

[12] Fan, Z., Li, J., Zhou, S., and Xu, B., "Speaker-Aware Speech-Transformer," 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 222-229.

[13] Jia Pan, Diyuan Liu, Genshun Wan, Jun Du, Qingfeng Liu, and Zhongfu Ye, "Online speaker adaptation for lvcsr based on attention mechanism," in Proceedings, APSIPA Annual Summit and Conference, 2018, vol. 2018, pp. 12–15.

[14] W. Hsu, Y. Zhang, and J. R. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in Proc. NIPS, 2017, pp. 1876–1887.

[15] W. Hsu and J. R. Glass, "Extracting domain invariant features by unsupervised learning for robust automatic speech recognition," in Proc. ICASSP, 2018, pp. 5614–5618.

[16] Siyuan Feng and Tan Lee, "Improving Unsupervised Subword Modeling via Disentangled Speech Representation Learning and Transformation," in Interspeech 2019.

[17] J. Chorowski, R. J. Weiss, S. Bengio and A. van den Oord, "Unsupervised Speech Representation Learning Using WaveNet Autoencoders," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 27, no. 12, pp. 2041-2053, Dec. 2019, doi: 10.1109/TASLP.2019.2938863.

[18] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass, "An unsupervised autoregressive model for speech representation learning," in Interspeech, 2019.

[19] Y. Chung and J. Glass, ”Generative Pre-Training for Speech with Autoregressive Predictive Coding,” ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 3497-3501, doi: 10.1109/ICASSP40776.2020.9054438.

[20] Yu-An Chung, Hao Tang and James Glass, ”Vector-Quantized Autoregressive Predictive Coding,” in Interspeech, 2020.

[21] I. Kalith, David Asirvatham, and Raisal Ismail, “ Context-dependent Syllable Modeling of Sentence-based Semi-continuous Speech Recognition for the Tamil Language,” 2017. Information Technology Journal. 10.3923/itj.2017.

[22] Lokesh, S., Kumar, P.M., Devi, M., Panchatcharam, P., and Babu, G.C. (2018). ”An Automatic Tamil Speech Recognition system by using Bidirectional Recurrent Neural Network with Self-Organizing Map,” Neural Computing and Applications, 31, 1521-1531.

[23] Akila, A. and Chandra, E. ”Performance enhancement of syllable based Tamil speech recognition system using time normalization and rate of speech,” CSIT 2, 77–84 (2014). doi: 10.1007/s40012-014-0044-6.

[24] A. Madhavaraj and A. G. Ramakrishnan, ”Design and development of a large vocabulary, continuous speech recognition system for Tamil,” 2017 14th IEEE India Council International Conference (INDICON), Roorkee, 2017, pp. 1-5, doi: 10.1109/INDICON.2017.8488025.