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

Penn
University of Pennsylvania

மாநாட்டுக் கட்டுரைகள்
CONFERENCE PAPERS

TAMIL INTERNET 2011

தமிழ் இணையம் 2011

Natural Language Processing

(இயற்கை மொழி பகுப்பாய்வு)

An Efficient Tamil Text Compaction System

N.M..Revathi, G.P.Shanthi, Elanchezhiyan.K, T V Geetha,
Ranjani Parthasarathi & Madhan Karky
Tamil Computing Lab (TaCoLa),
College of Engineering Guindy, Anna University, Chennai.
haisweety18@gmail.com, jijutodo@gmail.com, madhankarky@gmail.com

Abstract

Tamil is slowly becoming the online language and mobile text messaging languages for many Tamils around the world. Social networks and mobile platforms now extensively support Unicode and applications for keying Tamil text. The number of characters in a text message is limited in some social nets and mobile text messages. The need for compacting the text becomes essential as it translates to saving online storage space, cost and many more factors. The paper proposes a text compaction system for Tamil, a first of its kind in Tamil. The system proposed in this paper handles common Tamil words, acronyms/abbreviations and numbers. Morphological analyzer [1] and Morphological generator are used to stem inflexion words and replace them to compact using a mapping repository. The proposed work is tested with over 10,000 words and it is found that the final result is reduced to 40% of the original text. The paper concludes by discussing possible extensions to this system.

1. Introduction:

In all languages, using compact or short form of words in text messages, emails, and blogs is rapidly increasing. It is particularly popularly amongst young urbanities as it allows for voiceless communication, useful in noisy environment that would defeat a voice conversation and also buffered communication since the message the sender wants to convey can be accessed by the receiver at any time. Compacting text is thus necessary because of limited message length in blog sites and tiny user interface of mobile phone. Getting the shortest word has no rule and it is mainly aimed at understanding. That is, those words should be understood by everyone. We can obtain the compact words by omitting letters, replacing prefix and suffix of through suitable symbols and numbers. This causes the compacted system to be credited with creating a language. The paper proposes a Text Compaction system for Tamil, the primogenital in Tamil..

2. Background:

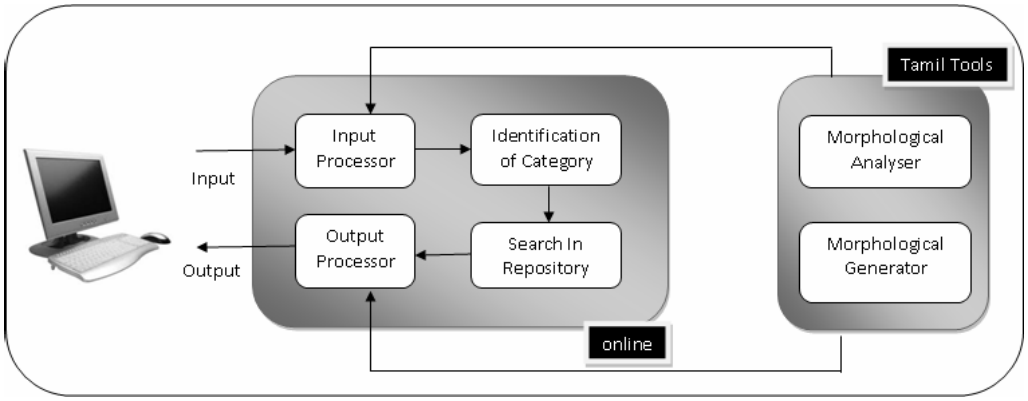
Tamil is perhaps the only classical language, whose glorious literatures date back to the pre-Christian era, has remained in continuous use for more than many millennia now. Due to the untiring efforts of scholars, researches and enthusiasts, it has also evolved creatively over the years to the extent that it is also used today profusely in computers, internet, mobile phone etc. Diverse creative efforts have been taking place that would pave the way for a quantum jump in the usage of Tamil in Information Technology. “Tamil Virtual University”, “Centre for Research and Applications of Tamil in Internet”,

"Tamil Software Development Fund" is to quote a few. These efforts paved the way for the motivation of proposing Tamil compaction system in Tamil.

Many compaction systems have been developed for English and other languages. Lee Ming Fung in [2] proposed a Short form Identification and Categorization model based on maximum entropy to identify short forms from actual words and acronyms/abbreviations and categorize the short forms into the short forms formed from letter omission and those formed through phonetic substitution of parts of words. In the proposed system the compact words are formed in a diverse variety of ways such as omission, truncation and phonetic substitution. Acronym Identification and detection has been much researched. Acrophile in [3] automatically searches acronyms from acronym-expansion pairs from domain specific databases. By acronyms expansion pairs, we refer to a pairs each containing acronyms and their full expanded form or meaning. The paper makes use of acronym expansion pairs to replace the full expanded form with the acronyms.

3. Text Compaction Framework:

The figure below presents the various components of the framework.



3.1 Input Processing

The input text is tokenized based on a delimiter and is passed on to the Morphological Analyzer. The analyzer removes the suffix (if present) added to the word and delivers the root word (RW). For example if the input to the analyzer is கணிப்பொறியில் the output is given as கணிப்பொறி.

3.2 Identification of the type

The proposed paper handles three categories of words; common Tamil words, Abbreviations /acronyms, numbers. Now, the category to which the RW belongs is to be identified. The RW is checked to decide the category of abbreviations /acronyms. This is done by comparing the root word with the keys of the hash map (2.3). If the comparison results are true then the RW is considered as the abnormal word (AW) i.e. it belongs to the category of acronyms/abbreviations, else, it is treated as the normal word (NW) i.e. it belongs to either the first or third category.

3.3 Extraction of the compact word

If the word is identified as a normal word, it is passed to a tree which is built dynamically from the set of words that has already been stored in the dictionary. The NW is then searched in the binary search tree. On finding the NW in the binary search tree, the compact word is retrieved with an efficient mapping algorithm that maps each of the normal word with its compact word.

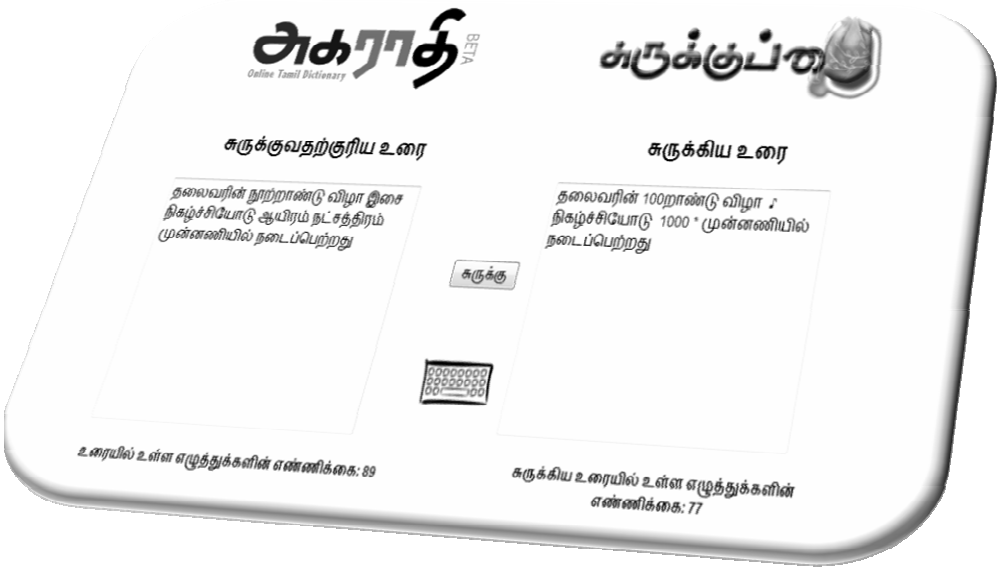
Say suppose the word is an abnormal word, its compact word is retrieved in the following manner. A linked hash map is built for all the abbreviated words. The hash map uses the first word the abbreviated word as its key. Again with the help of an efficient mapping algorithm, the compact word is retrieved. In case the NW is a number name it is replaced with the numerals based on the place value system.

3.4 Output Processing

The compact word that is being extracted is passed on the Tamil tool Morphological Generator to add the suitable suffix to cater to the rules of the language.

4. Results and Analysis:

The paper proposes the following layout for displaying the results to the user. It has two text areas: the one on the left is for entering the input text and the other on the right for displaying the output. The user can also view the no of characters that have been reduced in the output text.



Efficiency of the system can be calculated as (no of characters in the input text / no of characters in the output text) X 100%. The proposed work is tested with over 10,000 words and it is found that the final result is reduced to 40% of the original text.

5. Conclusion and Future work:

The paper describes the Tamil Compaction System, a framework for shrinking the text such that its meaning remains the same. Different subsystems and components of the framework are described in detail. Results from the implementation of this Tamil compaction system framework is provided and is compared against the compacting third party applications of social networking sites that are implemented for English language. Improving the mapping for words which are frequently used, conceptual reducing, integrating numerical analyser will take this system to its next level.

References:

- Anandan, R. Parthasarathi, and T.V. Geetha, *Morphological Analyser for Tamil*. ICON 2002, 2002.
- Fung, L. M. (2005). *SMS short form identification and codec*. Unpublished master's thesis, National University of Singapore, Singapore
- *Acrophile* (LSLarkey, P Ogilvie, MA Price, B Tamilio, 2000) a system that automatically searches acronym expansion pairs.
- *Short Message Service (SMS) Texting Symbols: A Functional Analysis of 10,000 Cellular Phone Text Messages* by Robert E. Beasley, Franklin College.

Tamil Summary Generation for a Cricket Match

J. Jai Hari Raju, P. Indhu Reka, K.K Nandavi, Dr. Madhan Karky
Tamil Computing Lab (TaCoLa),
College of Engineering Guindy, Anna University, Chennai.
jaihari1989@gmail.com, p.indhu@gmail.com, ashwathas@gmail.com, madhankarky@gmail.com

Abstract

Cricket is one of the most followed sports in the Indian subcontinent. There is a wide requirement for natural language descriptions, which summarize a cricket match effectively. The process of generating match summaries from statistical data is a manual process. The objective of this paper is to propose a framework for automatic analysis and summary generation for a cricket match in Tamil, with the scorecard of the match as the input. Data analytics is performed on the statistical match data, to mine all frequently occurring patterns. The paper proposes a parameter called ***Interestingness,*** which quantifies the interestingness of the match. The paper also proposes a customization model for the summary. We propose an evaluation parameter called ***humanness,*** which quantifies the similarity between the output and a manually written summary. Discussing the results and analyzing the summaries generated for matches based on scorecards, this paper concludes with proposing some extensions for future developments.

1. Introduction

The number of websites which facilitate people to follow and analyze sports has increased manyfold. Among them, there are an exceptionally large number of sites devoted to Cricket. Mostly these involve participation of experts, who present their views and summaries in English about cricket matches. There are no such sites which provide similar services in Tamil. In this case it is also desirable if there is an alternative for human creativity. As a solution the paper proposes an automated Tamil summary generation framework which is capable of analyzing and generating a Tamil summary about a cricket match, provided the score card as the input. This paper discusses the overall architecture and implementation details of such a framework.

The large amount of data in this domain makes it possible to apply data mining and data analytics techniques. The input scorecard is analyzed to construct feature vectors, which are then subjected to data mining. Based on the various parameters identified, the interestingness of the match is quantified.

The summary generation part involves extraction of key players and events from a match. Appropriate sentences are then synthesized to express these selected events. The sentence constructs and the vocabulary used are chosen based on the linguistic ability specified by the user. Then the sentences are combined in to a meaningful summary.

The results of the system, i.e. the summaries, are evaluated based on the ***Humanness*** parameter. This parameter gives the degree of similarity between the generated summary and the manually written summary, with which it is compared. This value helps us decide, the level of creativity achieved by

the system. In section 2 we provide an overview of the literature survey conducted. In section 3 we discuss the design of the various modules of the framework. In section 4 we discuss the implementation of the proposed framework and the results obtained from the analysis. Finally we conclude in section 5 with extensions to the current framework and directions for further studies in the field of Tamil Summary Generation Systems.

2. Background

In the literature there are existing works on summary generation from statistical data. Alice Oh et al. generated multiple stories about a single baseball game based on different perspectives using a reordering algorithm [1]. Ehud Reiter et al. in their book building natural language generation systems explain the difference between natural language generation and natural language processing and also describe the various steps involved in the natural language generation process with examples [2].Jacques Robin et al. presented a system (called STREAK) for summarizing data in natural language. It focuses on basketball game to design and evaluate the system [3]. L. Bourbeau et al. came up with the FoG (Forecast Generator) using the streamlined version of the Meaning-Text linguistic model. This system was capable of generating weather forecasts in both English and French [4].

3. Summary Generation Framework

The Tamil Cricket Summary Generator consists of the following major components:

- Data Gathering and Modeling module
- Data Mining and Data Analytics module
- Summary Generator
- Evaluator

Figure (1) given below depicts the Summary Generation framework.

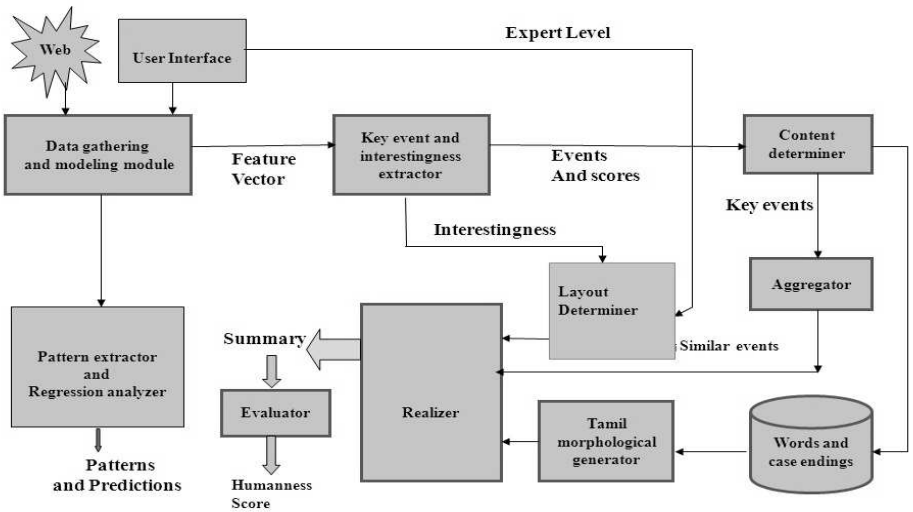


Figure 1: Tamil Cricket Summary Generator Framework

3.1 Data Gathering and Modeling Module

Data gathering is the first step of the system. The data to be gathered is present in internet. This module has a custom designed parser, for the tag structure of the site. The user must provide the URL from where the particular match's data can be obtained. The module checks whether the match has already been processed. If not the parser parses the page and retrieves the statistical data. The statistical data is then modeled in the form of the predefined feature vectors.

3.2 Data Mining and Analytics Module

Modified version of Apriori algorithm is used to find the association rules from the feature vectors. After performing mathematical analysis using correlation of variance (CoV), CoV is plotted against average to give an idea about how consistent the player is. The interestingness of the match is calculated based on the weighted average of the scores assigned to the factors identified, they include the Winning margin, Team history, Individual records made, High run rate, Series state, Relative position in international ranking, Reaction in social networks etc.

3.3 Summary Generator

The summary generator part of the framework consists of the following sub modules Content Determiner, Aggregator, Tamil Morphological Generator and Layout Determiner. The events to be included in the summary are not predefined and are not the same for every match. Based on the interestingness of the total match, the interestingness of the individual events and the expert level chosen by the user, particular events are chosen to be included in the summary. The content determiner is responsible for identifying those facts which are worth mentioning in the summary.

Aggregation of relevant events from other matches in the summary will make it more readable and interesting. The aggregator performs this function. It chooses events based on their similarity and coherence and aggregates them with the key events selected in the content determiner module.

As a next step, the sentences used to describe the events are synthesized. The sentence which is the most apt to the current event under consideration is selected. The vocabulary used in the sentence and the depth to which an event is discussed is also varied based on the expert level of the user. The nouns in the key events are passed to the morphological generator along with the desired case endings and the generated variants are added to the sentences.

The layout determiner module chooses the layout of the summary to be generated. The layout is varied based on the interestingness of the match. The sentences are aggregated in the fashion of the layout selected and the final output summary is passed to Evaluator.

3.4 Evaluator

The summary generated by the system is evaluated based on its degree of similarity with human written summaries. The summaries are compared based on two parameters, the Nouns Mentioned and the Events Mentioned.

The nouns and the events in the summaries are extracted along with their absolute positions. The events in the summary are modeled as a set consisting of, one or more Performers (the persons who takes part in the event), Numeral (the numeric part involved in the event e.g. 4 wickets) and a

Descriptor (the action connecting the Performer and the Numeral). Their absolute positions refer to the sentence number in which they are mentioned. Then these absolute positions are normalized based on the total number of sentences present in the summary. Three different scores are calculated they are,

- Similarity Score: The ratio of the number of nouns and events mentioned in both the summaries to the total number of nouns and events mentioned at least in one summary.
- Count Score: The ratio of the number of nouns and events mentioned in the system generated summary to the number of nouns and events mentioned in the human written summary
- Closeness Score: The degree of closeness, in terms of the normalized positions of the nouns and events mentioned in both the summaries.

A weighted average of these three scores yields the final humanness score.

4. Implementation

To implement the proposed framework, *espncricinfo* a reliable and prominent site for Cricket data is chosen as the source of input. The frame work was implemented in java. The URL of the match for which the summary is to be generated is obtained from the user. The feature vectors designed for modeling a match are stored as rows with unique identities, in the back end oracle database. The patterns which are generated as a result of data mining are validated based on the support and confidence parameters. As a design decision all nouns are stored in English and are translated on the fly using a constantly updated look up database. This decision was taken to allow interoperability and easy extension of the system to other languages in future. The sentence pattern files are stored external to the system, so as to allow modifications without changes in the system. The summary generated for the match is stored in the back end, indexed with the unique identity assigned already. The user interface is designed to be simple and robust. It allows the users to search matches based on various parameters and also to save their preferences.

4.1 Results

Score cards of 90 One Day International matches where retrieved and their summaries were generated. These include matches between 9 countries. Both individual matches and series were considered. A large number of hidden patterns in cricket domain have been retrieved based on the algorithm used. The patterns have been validated and the ones which are interesting have been reported. The factors contributing to the interestingness of the match have been identified and the weights associated with them have been found. The consistency of a player has been modelled and consistency analysis of a player is done to analyse his performance.

The difference in the language used and the events mentioned in the summary is pronounced when the user opts for an expert level. Similar facts occurring in the past have been identified and added to the summary. Each summary was compared with two human written summaries, one an expert summary and other an average summary, their cumulative scores were considered. The humanness score of the summaries tend to be in the range of 70% to 85%. The recurrence of layouts is also minimal, which reflects the fact that the summaries generated are not monotonous.

5. Conclusion and future work

In this paper we have proposed the framework for an Automated Tamil Cricket Summary Generator. The current implementation of the system can be enhanced by adding machine learning capabilities to make the summaries more human and interesting. The system can be extended to produce summaries in multiple languages apart from Tamil. The system can be enhanced to generate summaries about the match in real time. As a next level the system can be modified for summary generation in other sports too.

Figure 2: Screenshot of the Tamil Cricket Summary Generation System

The frame work can be used as a guideline to develop summary generation systems, which can be applied for any domain where frequent numerical reports are used. (Weather Prediction, Industrial Quality Testing etc)

References

- Alice Oh and Howard Shrobe, "Generating baseball summaries from multiple perspectives by reordering content," in Proc. 5th International Natural Language Generation Conference, 2008, pp. 173-176.
- Ehud Reiter and Robert Dale, "Building natural language generation systems," Cambridge: Cambridge University Press, 2000.
- Jacques Robin and Kathleen McKeown, "Empirically Designing and Evaluating a New Revision-Based Model for Summary Generation," Department of Computer Science, Columbia University, 1996, vol. 85, pp.135-179.
- L. Bourbeau, D. Carcagno, E. Goldberg, R. Kittredge and A. Polguere. "Bilingual generation of weather forecasts in an operations environment," In Proc. 13th International Conference on Computational Linguistics, Helsinki University, Finland, 1990. COLING

Lyric Mining: Word, Rhyme & Concept Co-occurrence Analysis

Karthika Ranganathan, T.V Geetha, Ranjani Parthasarathi & Madhan Karky

Tamil Computing Lab (TaCoLa),

College of Engineering Guindy, Anna University, Chennai.

karthika.cyr@gmail.com, madhankarky@gmail.com

ABSTRACT

Computational creativity is one area of NLP which requires extensive analysis of large datasets. Laalalaa [1] framework for Lyric analysis and generation proposed a lyric analysis subsystem that required statistical analysis of Tamil lyrics. In this paper, we propose a data analysis model for words, rhymes and their usage in Tamil lyrics. The proposed analysis model extracts the root words from lyrics using a morphological analyzer [2] to compute the word frequency across the lyric dataset. The words in their unanalyzed form are used for computing the frequent rhyme, alliteration and end-rhyme pairs using adapted apriori algorithm. Frequent co-occurring concepts in lyrics are also computed using Agaraadhi, an on-line Tamil dictionary. Presenting the results, this paper concludes by discussing the need of such an analysis to compute freshness, pleasantness of a lyric and using these statistics for Lyric Generation.

Keywords : Tamil Lyrics, Morphological Analyser, Apriori algorithm.

I. INTRODUCTION

Tamil is one of the world's oldest languages and has a Classical status. Numerous forms of literature exist in Tamil language of which, lyrics play a vital role in taking the language to every house hold in form of original film soundtracks, jingles, private albums, and commercials. With over thousands of lyrics being created every year, we do not have proper tools to model and analyse lyrics. Such an analysis framework would enable one to see various patterns of words, combinations and thoughts used over time. The analysis framework will also make it possible to generate fresh lyrics where the freshness can be associated with the concepts and thoughts associated with the lyric.

In this paper, we discuss about Tamil lyric Analysis on the basis of word usage, rhyme usage and the co-occurrence of word. The frequency of word usage is identified by considering a morphological root of the word using morphological analyser, instead of considering terms. For analysing the frequent rhyme, alliteration and end-rhyme pairs, we adapted Apriori algorithm [4]. To identify the co-occurring concepts in lyrics, we used "Agaraadhi", an on-line Tamil dictionary Framework [3] and a new algorithm has been proposed to compute the frequent usage of co-occurring concepts in lyrics.

The rest of this paper has been organized as follows. In Section 2, we explain about the algorithm and tools. In Section 3, we explain the methodology and in Section 4, we discuss our results. Conclusions and future extensions to this work are presented in section 5.

2. MORPHOLOGICAL ANALYSIS AND APRIORI ALGORITHM

Morphological analysis is the process of segmenting words into morphemes and identifying its grammatical categories. For a given word, morphological analyser (MA) generates its root word and its grammatical information. The role of morphological analyser in the proposed work is to identify the noun and verb morphology of a given word, examples are as follows

Example 1, for noun morphology:

இராமனை (Ramanai)

இராமன் (Raman) + ஐ (ai)

Entity + Accusative Case

Example 2, for verb morphology:

சென்றான் (**Senraan**)

செல் (sel) + ற் (R) + ஆன் (Aan)

Verb + Past Tense Marker + Third Person Masculine Singular Suffix

In the proposed work, the frequency of a word is identified by considering the variations of a noun in terms of its morphology. For instance, the variations of Ramanai such as Ramanaal, Ramanukku, Ramanin, Ramanadhu are also counted for the word Raman.

The Apriori Algorithm is an influential algorithm for mining frequent item sets for Boolean association rules [4]. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. It has objective measures: support and confidence. The support of an association pattern is the percentage of task-relevant data transactions for the apparent pattern. Confidence can be defined as the measure of certainty or trustworthiness associated with each discovered pattern.

3. LYRIC ANALYSIS

(i) Word Analysis

The frequency of words is used to associate a popularity score for each word. This score is proposed to be used for lyric generation part of the frame work proposed in [1]. In this work, the popularity score of a word has been identified from lyrics. In lyrics, the words are mainly attached with the suffix. So, the root words are taken into consideration for determining its frequency count. The root words are identified using morphological analyser. The algorithm to find the word usage is illustrated below:

Algorithm

Let L_D is the set of lyric dataset and L_S denotes the set of sentences of lyric dataset and L_W denotes the set of words in the lyric dataset. W_C denotes the word count across all lyric dataset. Let m be the total number of sentences in lyrics and n be the total number of words in lyrics.

a) Given a Lyric dataset L_D

b) For each $L_{Si} \leftarrow 1$ to m

Split the sentence L_S into words L_W

c) For each $L_{Wj} \leftarrow 1$ to n

$R_W \leftarrow$ ProcessMorphAnalyser(L_{Wj})

d) Let R_W be the root word

if R_W exist, then add into the word count list (W_C)

else add into the word count list (W_C)

e)ReturnW_C.

Here ProcessMorphAnalyser(L_{Wj}) returns the root of the given word.

(ii) Rhyme Analysis

Alliteration (Monai) is the repetition of the same letter at the beginning of words**.** The rhyme (Edhugai) is defined as the repetition of the same letter at the second position of words. The end rhyme (iyaibu) is defined as the repetition of the same letter at the last position of words. The example of alliteration, rhyme and end rhyme is given below:

Examples:

உயிர் and **உன்** rhyme in alliteration (monai) as they start with the same letter.

இதயம் and **காதல்** rhyme in rhyme (edhugai) as they share the same second letter.

யாக்கை and **வாழ்க்கை** rhyme in end – rhyme (iyaibu) as they share the same last letter.

We have adapted apriori algorithm to find the frequency count of rhyme, alliteration and end rhyme pairs of Tamil lyrics which has been illustrated below:

Algorithm:

Let L_D be the set of lyric dataset and L_S denote the set of sentences of Lyric dataset. Let m be the total number of sentences in lyrics. Let P_{C1} denote the count of alliteration and P_{C2} denote the count of rhyme and P_{C3} denote the count of end – rhyme.

a) Given lyric dataset L_D .

b) For each $L_S \leftarrow 1$ to m

Join the pair of sentences (L_P)

c) For each L_P

Consider the first (L_{P1}) and last words (L_{P2})

d) Rhyme ($L_{P1, LP2}$)

e) return P_C

Algorithm : Rhyme ($L_{P1, LP2}$)

a) Let k denote the i_{th} character. Let M_L denote the alliteration (monai) list and R_L denote the rhyme (edhugai) list and E_L denote the end – rhyme (iyaibu) list.

b) For ∀i,

if k = 1, if $L_{P1(k)} = L_{P2(k)}$, then add into M_L list and increment P_{C1}

if k = 2, if $L_{P1(k)} = L_{P2(k)}$, then add into R_L list and increment P_{C2}

if k = i - 1, if $L_{P1(k)} = L_{P2(k)}$, then add into E_L list and increment P_{C3}

(iii) Co-occurrence concept Analysis

Co-occurrence is defining the frequent occurrence of two terms from a text corpus on the either side in a certain order. This word information in NLP system is extremely high. It is very important for cancelling the ambiguous and the polysemy of words to improve the accuracy of the entire system [5].

In this method, to improve the efficiency of co-occurrence, we have been considering the concept of each word. The concept for each word has been identified using the Agaraadhi, an on-line Tamil dictionary. The example for concept word which has been in lyric is given below:

Example: The word "**நிலவு** "which has the concept **வெண்ணிலா**, **மதி**, **மாதம்**, **துணைக்கோள்**, **வெண்ணிலவு**, **அம்புலி**, **அம்புலிமான்**.

By considering these concepts, we have been determining the co-occurring words using our own algorithm is described below:

Algorithm :

Let L_D denote the set of Lyric dataset and W denote each word in lyric. Let C_W denote the set of concepts for each word. Let W_C denote the word count.

a) If the word C_W identify, then consider the next word.

Increment the count W_C

b) Else the word W and consider the next word.

Increment the count W_C

c) return W_C

4. RESULTS

The lyric corpus of more than two thousand songs were analysed for the word usage, rhyme usage and Co-occurence concepts usage. The analysed results are given below:

Table 1 shows the list of top 10 usage words in lyrics.

WORDS	USAGE	WORDS	USAGE
நீ	2009	வா	1062
என்	1941	ஒரு	987
நான்	1645	கண்	965
உன்	1556	பூ	857
காதல்	1153	இல்லை	793

Table 2 shows the list of top 10 rhyme words in lyrics.

EDHUGAI	USAGE	MONAI	USAGE	IYAIBU	USAGE
என்,உன்	107975	என்,என்னை	33492	என்,உன்	111552
நான்,என்	80125	உன்னை,உன்	26289	நான்,என்	83435
நான்,உன்	61204	எந்தன்,என்	16478	நான்,உன்	63543
என்,உன்னை	33731	என்,என்ன	15405	என்,உந்தன்	18411
என்,என்னை	32570	உந்தன்,உன்	14001	எந்தன்,என்	16478
உன்னை,உன்	25747	உயிர்,உன்	11640	உந்தன்,உன்	14001
என்னை,உன்	24867	இந்த,இது	10993	எந்தன்,உன்	12524
நான்,உன்னை	19297	எனது,என்	9985	நான்,உந்தன்	10524
நான்,என்னை	18935	எந்த,என்	9976	எந்தன்,நான்	9367
என்,என்ன	14486	நீ,நீயும்	9962	இந்த,அந்த	4818

Table 3 shows the list of top 10 co-occurring concept words in lyrics.

CO-OCCURING WORDS	USAGE
அன்பே,அன்பே	638
சின்ன,சின்ன	530
வா,வா	506
என்,காதல்	478
ஒரே,ஒரு	469
நீயும்,நானும்	434
உன்னை,நான்	419
தமிழ்,எங்கள்	367
ஒரு,நாள்	302
நீ,என்னை	287

By analysing those data, this shows that the most of lyrics which predicts the emotion of happiness and love. In the adapted apriori algorithm, the support which represents the total number of pair words with the total number of combination of sentences and the confidence which described the total number of pair words with the total number of pair of sentences. The results may vary if the number of lyrics used for the analysis is increased.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we discussed the usage of words and rhymes in the lyrics dataset. The adapted apriori algorithm has been used to detect the frequency count for rhyme, alliteration and end word pairs. This analysis has been mainly used in the lyric generation and computing freshness scoring for lyrics. Frequent co-occurring concept is also been identified for the development of lyric extraction, semantic relationship, word sense identification and sentence similarity. Possible extensions of this work could be the detection of emotions in lyrics by genre classification and identify genre specific rhymes and concept co-occurrence.

REFERENCES

- Sowmiya Dharmalingam, Madhan Karky, "LaaLaLaa – A Tamil Lyric Analysis and Generation Framework" in World Classical Tamil Conference – June 2010, Coimbatore.
- Anandan P, Ranjani Parthasarathy, Geetha, T.V. "Morphological analyzer for Tamil". ICON 2002.
- Agaraadhi Online Tamil Dictionary. http://www.agaraadhi.com, Last accessed date 25h April 2011.
- HAN Feng, ZHANG Shu-mao, DU Ying-shuang, "The analysis and improvement of Apriori algorithm", Journal of Communication and Computer, ISSN1548-7709, USA, Sep. 2008, Volume 5, No.9 (Serial No.46).
- EI-Sayed Atlam, Elmarhomy Ghada, Masao Fuketa, Kazuhiro Morita and Jun-ichi Aoe, "New Hierarchy Technique Using Co-Occurrence Word Information", International Journal of Information Processing and Management, Volume 40 Issue 6, November 2004.

Template based Multilingual Summary Generation

Subalalitha C.N, E.Umamaheswari, T V Geetha,
Ranjani Parthasarathi & Madhan Karky
subalalitha@gmail.com
Tamil Computing Lab (TaCoLa)
College of Engineering Guindy Anna University, Chennai.

Abstract

Summarization of large text documents becomes an essential task in many Natural Language processing (NLP) applications. Certain NLP applications deal with domain specific text documents and demand for a domain specific summary. When the essential facts are extracted specific to the domain, the summary proves to be more efficient. The proposed system builds a bilingual summary for an Information Retrieval (IR) system named CoRee, which tackles Tamil Language and English Language text documents [1]. As the input documents are tourism domain specific documents, the summary is extracted based on specially designed seven tourism specific templates 7 both for Tamil and English. The templates are filled in with the required information extracted from the UNL representation and a bilingual summary is generated for each text document irrespective of the language of input text document. The efficiency of the summary has been tested manually and it has achieved 90% efficiency. This efficiency depends on factors other than summary generation such as enconversion accuracy and dictionary entry coverage. The proposed system can be extended for many languages in future.

1. Introduction

Automatic summary generation has been a research problem for over 40 years [2]. Summarizing the texts helps in avoiding information overload and also saves time. Multi lingual Natural Language applications have emerged in great number in recent years. This makes the need for a multi lingual summary generation a quintessential task. Alkesh patel et al have come up with a multi lingual summary generation by using structural and statistical factors [2]. David Kirk Evans has generated multi lingual summary using text similarities existing in the sentences [3]. Dragomir Radev et al have developed a multi lingual summary generation tool named MEAD using centroid and query based methods. They have also used many learning techniques such as decision trees, Support Vector Machines (SVM) and Maximum Entropy [4].

All the above works on multi lingual summarization have not used a interlingua document representation . We propose that a multi lingual summary can be generated with much more ease by using a interlingua document representation language called, "Universal Networking Language" (UNL) [5]. UNL converts every term present in a natural language text document into a language independent concept, thereby making the applications built using it a language independent one. The proposed work extracts a domain specific summary, as the UNL documents used are tourism domain specific. Tourism specific templates are framed and the sentences fitting the templates are chosen and formed as a summary.

The rest of the paper is organized as follows. Section 2 gives a brief introduction about UNL. Section 3 describes the proposed summarization technique. Section 4 discusses the evaluation of the proposed work. Section 5 reveals the enhancements needed to the proposed work and Section 6 gives the conclusion of the paper.

2. Universal Networking Language

UNL is an intermediate language that processes knowledge across language barriers. UNL captures the semantics of the natural language text by converting the terms present in the document to concepts. These concepts are connected to the other concept through UNL relations . There are 46 UNL relations like plf(Place From), plt(Place To), tmf(Time from), tmt(Time to) etc [1]. This process of converting a natural language text to UNL document is known as Enconversion and the reverse process is known as Deconversion. The UNL document is normally represented as a graph where the nodes are concepts and edges are UNL relations. An example UNL graph is shown for the example 1.

Example 1: John was playing in the garden .

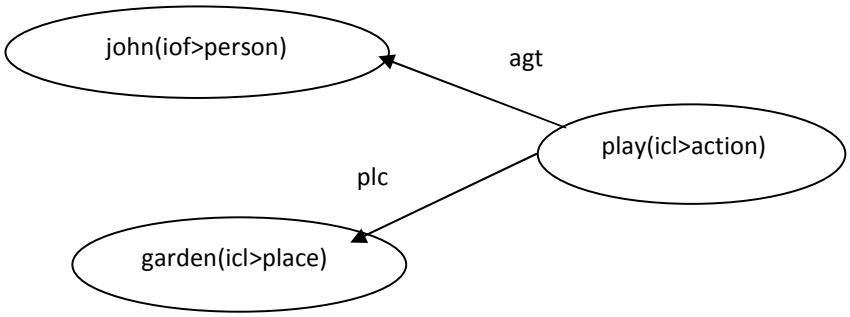


Figure 1: UNL graph for Example 1

The nodes of graph namely, "John(iof>person)", "Play(icl>action)" and garden(icl>place) represent the terms John, playing and garden present in the example 1. The semantic constraints in the concepts, "iof>person", "icl>action" and "icl>place" denotes the context in which the concepts occur. The edges namely, "agt" and "plc" indicates that, the concepts involved are agents and place. From the above discussion, it is shown that the UNL inherits many semantic information from the natural language text and portrays in a language independent fashion.

The proposed work uses Tamil language text documents and English language documents enconverted to UNL for summary extraction which is described in the next section.

3. Template based Information Extraction

As discussed earlier, the summary is generated using the tourism specific templates. Figure 2 shows the over view of the proposed summary generation framework. The Framework consists of both language dependent and independent parts. The functionalities involving UNL are language independent and the inputs supplied to the framework to generate bi lingual summary are the language dependent parts. The bilingual summary generation is explained in the coming sections.

The seven templates describe about the tourism specific information of a place such as,god, food, flaura and fauna, boarding facility, transport facility, place and distance. The correct information for these templates are extracted as discussed below. The usage of semantics helps greatly in eliminating the ambiguities that may arise while picking up a concept to fill the slot. For instance, the word, "bat" may denote a cricket bat or the mammal bat.

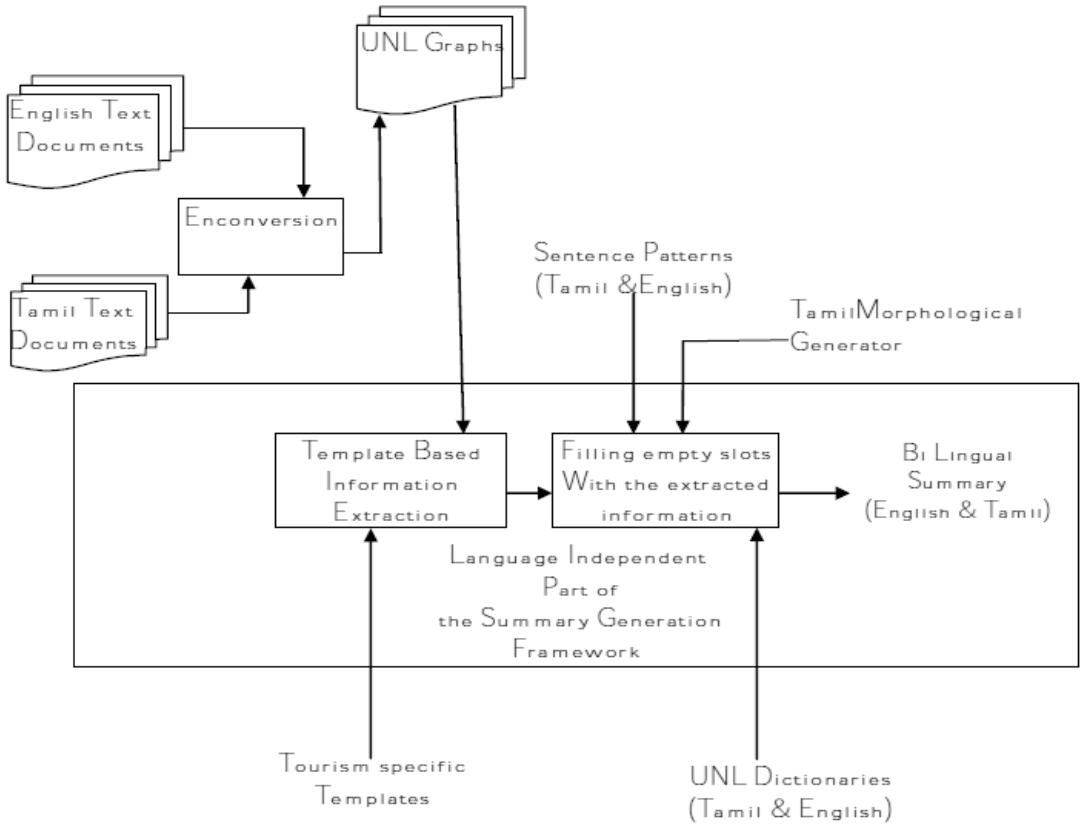


Figure 2: Overview of the Summary Generation Framework

This type of ambiguity is resolved by the semantic constraint, as the cricket bat will get the semantic constraint, "obj<thing (object thing)", whereas the mammal gets the semantic constraint, "icl>mammal". **Table 1** displays few semantics used for the respective templates.

The extracted tourism specific concepts are converted to the target language terms for building a summary using the sentence patterns which is explained in the next section.

4 Multi Lingual Summary generation

The information (concepts) extracted from the UNL graph using the templates are converted to the target language term using the respective UNL dictionary. For instance, to generate the English summary, the concepts comprising the semantic constraints are converted to English terms using the English UNL dictionary which consists of mapping between English terms and UNL concepts. These terms which when filled into the appropriate English sentence patterns, gives a English summary . The same procedure is done for building a Tamil summary. For each UNL graph irrespective of its source language, a summary in Tamil and English are generated.

Template	Semantics
God	iof>god, iof>goddess, icl>god
Food	icl>food, icl>fruit
Flaura and Fauna	icl>animal, icl>reptile, icl>mammal, icl> plant
Boarding facility	icl>facility
Transport facility	icl>transport
Place	icl>place, iof>place, iof>city, iof>country
Distance	icl>unit , icl>number

Table 1 :Semantics used for each templates

The terms obtained from the UNL dictionary will be a root word. For instance, the term, "eating" will be entered as eat (icl >action) in the UNL dictionary. So the terms obtained from the UNL dictionary needs to be generated to its original form using Morphological generator. The summary generation requires only tourism specific concepts, so the generation is almost not required . But we have used a morphological generator for Tamil, as the place information and distance information in Tamil with the case suffixes இல் (il), இலிருந்து(ilirunthu), உக்கு (ukku) etc needs to be generated. For the example UNL graph shown in figure 3, the generated transport template in Tamil which is part of the summary is given in example 2.

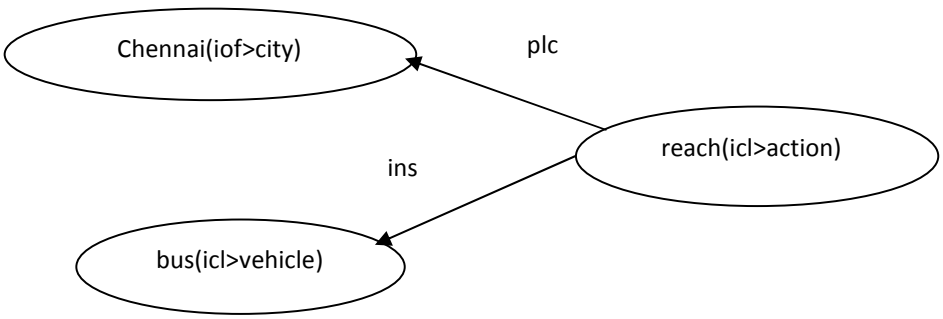


Figure 3:UNL graph given as input for example 2

Example 2: சென்னைக்கு பேருந்தில் செல்லலாம்

The concept chennai(iof>city) in the above graph, is generated as "சென்னைக்கு" by adding the case suffix "உக்கு " and the concept bus(icl>vehicle) is generated as

"பேருந்தில்" by adding the case suffix, "இல்".

5. Performance Evaluation

The proposed work has been tested with 33,000 Tamil and English text documents enconverted to UNL graphs. The performance of the methodology proposed has been evaluated using human judgement. The accuracy of the summary generated has achieved 90% . Apart from the summary generation factors such as tourism specific concept extraction , the accuracy also depends on the quality of enconversion and dictionary entry. By improving these factors, the accuracy can further be improved.

6. Conclusion and Future work

The proposed work generates a tourism specific bilingual summary using the intermediate document representation, UNL and tourism specific templates. The bilingual summary is generated in a simple and efficient manner compared to the earlier work done for multi lingual summary generation. The only over head involved is developing a enconverter framework.

As future enhancements, sentence patterns can be replaced by selecting the sentences having high sentence score based on its sentence position and the frequency of concepts. Query specific summary can also be generated on line, as the summary discussed here is a tourism specific generated off line using the templates. The evaluation of the generated summary can also be done by comparing it with the human generated summary. By doing this, many factors to make the machine generated summary compatible with human generated summary may evolve.

Reference

- Elanchezhiyan K, T V Geetha, Ranjani Parthasarathi & Madhan Karky, CoRe – Concept Based Query Expansion, Tamil Internet Conference, Coimbatore, 2010.
- Alkesh Patel , Tanveer Siddiqui , U. S. Tiwary , "*A language independent approach to multilingual text summarization*", Conference RIAO2007, Pittsburgh PA, U.S.A. May 30-June 1, 2007
- David Kirk Evans, "I*dentifying Similarity in Text: Multi-Lingual Analysis for Summarization* ", Doctor of Philosophy thesis, Graduate School of Arts and Sciences , Columbia University, 2005
- Radev, Allison, Blair-Goldensohn et al (2004), *MEAD – a platform for multidocument multilingual text summarization*
- The Universal Networking Language (UNL) Specifications Version 3 Edition 3, UNL Center UNDL Foundation December 2004.
- Jagadeesh J, Prasad Pingali, Vasudeva Varma, " *Sentence Extraction Based Single Document Summarization*" Workshop on Document Summarization, March, 2005, IIIT Allahabad.
- Naresh Kumar Nagwani, Dr. Shrish Verma , "*A Frequent Term and Semantic Similarity based Single Document Text Summarization Algorithm* " International Journal of Computer Applications (0975 – 8887) Volume 17– No.2, March 2011 .
- Prof. R. Nedunchelian, "*Centroid Based Summarization of Multiple Documents Implemented using Timestamps* " First International Conference on Emerging Trends in Engineering and Technology, IEEE 2008

Special Indices for LaaLaLaa Lyric Analysis & Generation Framework

Suriyah M, Madhan Karky, T V Geetha, & Ranjani Parthasarathi
{suriyah.cse@gmail.com, madhankarky@gmail.com,
tv_g@hotmail.com, rp@annauniv.edux}
Tamil Computing Lab (TaCoLa),
College of Engineering Guindy, Anna University, Chennai.

Abstract

With the advent of computational tools for creativity, it becomes inevitable to design data structures which cater to the specific needs of the creative form considered. A lyric generator has to retrieve words fast based on Part of Speech and rhyme. This paper aims at building special indices for the LaaLaLaa Lyric Generator framework based on POS and rhyme to facilitate faster retrieval. The retrieval times of the proposed model and the conservative word indexed model are compared. The indexing is based on the KNM(Kuril, Nedil, Mei) pattern and the letters that occur in the rhyming spots of the words. The data structure is organized as hash tables to ensure best retrieval complexity. Separate hash tables for each POS and rhyming scheme are created and populated. Here, the key would be the meter pattern with the letters occurring at the rhyming spots and the value would be the list of all those words which fall under the key's constraint. When the word indexed and meter rhyme indexed retrievals were compared, the latter reduced the average retrieval time drastically. There were not steep variations in the retrieval times as was in the former approach. This remarkable efficiency was traded-off with space.

1. Introduction

Tamil, one of the oldest languages, has a very rich literary history dating back to two thousand years. We have more than two thousand lyrics being written in this language in the form of film songs, advertisements, jingles, private albums etc. With the advent of computational tools for creativity, it becomes inevitable to design data structures which cater to the specific needs of the creative form considered. Tools for poem generation, story generation and lyric generation have been proposed.

A poem has to have three qualities – meaningfulness, poeticness and grammaticality [1]. A lyric is a poem which has constraints of having to satisfy a tune and a theme. This paper talks about building special indices for the LaaLaLaa lyric generator framework[2] to aid faster retrieval of words based on POS and rhyme.

A lyric generator would require the retrieval of words of a particular meter and particular letters at rhyming spots. This would maximize the poeticness of the lyric generated with the increase in rhymes. This retrieval process, when carried out on an un-indexed word database, is too expensive as it would take separate processing for meter, and each of the three rhymes in Tamil. To facilitate faster retrieval of words satisfying these constraints, the word database has to be indexed based on

meter-pattern and rhyme. This makes indexing of the word database based on the abovesaid constraints necessary.

This paper is organized as eight sections. The second section discusses about a few existing works in this area. The third section gives an overview of the Rhyme schemes in Tamil. An overall view of the system is given by the fourth section while the fifth section talks about the approach proposed for indexing. This work concludes with the results obtained and scope for future work in this area.

2. Background

Though significant number of works has been done in the arena of poetry generation in other languages, there is only less number in Tamil.

The "Automatic Generation of Tamil Lyrics for Melodies" [3] identifies the required syllable pattern for the lyric and passes this to a sentence generation module which generates meaningful phrases that match the pattern. This system generates rhyme based on maximum substring match and fails to make use of the three rhyming schemes that are specific to Tamil language. "LaaLaLaa - A Tamil Lyric Analysis and Generation Framework" [2] generates Tamil lyrics for POS tagged pattern with words from a rhyme finder according to rhyming schemes in Tamil. Nichols et al[5] investigate the assumption that songwriters tend to align low-level features of a song's text with musical features. K. Narayana Murthy[4] suggests having a non-dense TRIE index in main memory and a dense index file stored in secondary memory.

Anna Babarczy et al[6] suggested a hypothesis that a metaphoric sentence should include both source-domain and target-domain expressions. This assumption was tested relying on three different methods of selecting target-domain and source-domain expressions: a psycholinguistic word association method, a dictionary method and a corpus-based method. Hu, Downie and Ehmann[7] examine the role lyric text can play in improving audio music mood classification. Mahedero et al[8] argue that a textual analysis of a song can generate ground truth data that can be used to validate results from purely acoustic methods. Mayer et al[9] present a novel set of features developed for textual analysis of song lyrics, and combine them with and compare them to classical bag-of-words indexing approaches and results for musical genre classification on a test collection in order to demonstrate our analysis. "Semantic analysis of song lyrics" studies the use of song lyrics for automatic indexing of music. Netzer et al explore the usage of Word Association Norms (WANs) as an alternative lexical knowledge source to analyze linguistic computational creativity. Logan et al. use song lyrics for tracks by 399 artists to determine artist similarity[12].

3. Rhyme Schemes and Rhyme Patterns

Rhyme Schemes: English has number of rhyme effects like assonance, consonance, perfect, imperfect, masculine, feminine etc. This arises from the variation in stress patterns of words, lack of clear cut description about the spots where rhymes can occur.

In Tamil, the grapheme and phoneme are bound stronger than in English. There are 3 characteristic rhyme schemes in Tamil – Monai (மோனை), Edhugai (எதுகை) and Iyaibu (இயைபு).

Two words are said to rhyme in monai if their first letters are the same, in edhugai if their second letters are the same and in iyaibu if their last letters are the same.

Examples: பறவை and பச்சை rhyme in monai as they start with the same letter.

அருவி and விருப்பு rhyme in edhugai as they share the same second letter.

யாக்கை and வாழ்க்கை rhyme in iyaibu as they share the same last letter.

As one may infer, two words can rhyme in more than one pattern also.

Examples: அருவி and குருவி rhyme in edhugai and iyaibu.

கவிதைகள் and கவிஞர்கள் rhyme in all the three schemes.

Meter Pattern: One way of classifying alphabets of the Tamil language is based on the time interval (மாத்திரை - maathirai) for which they are pronounced. One maathirai corresponds to the time taken to wink the eyelid. The types of letters in this classification are

Nedil (N) (நெடில்) - Those alphabets which are pronounced for the time interval of 2 maathirai.

Kuril (K) (குறில்) - Alphabets which take 1 maathirai to be pronounced.

Mei (M) (மெய்) - Alphabets which are pronounced for 0.5 maathirai.

Meter pattern of a word refers to its Kuril Nedil Mei pattern.

For example, the meter pattern of the word பாடல் is NKM as பா is a Nedil(N), ட is a Kuril(K) and ல் is a Mei(M).

The indexing methodology proposed needs to take care of both meter pattern and rhyme to facilitate faster retrieval of words for the LaaLaLaa Lyric Generation and Analysis framework[2].

4. Overview of the System

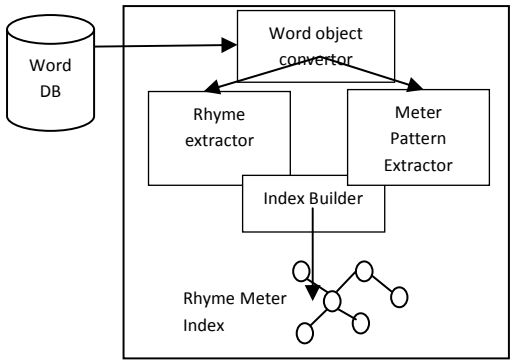


Figure 1. Overview of the system.

Each word from the word database is converted to an object. Meter pattern and the alphabets at the Rhyming positions of each word are found out by Meter Pattern Extractor and Rhyme Extractor respectively. Using the data obtained, the Index builder builds the Rhyme Meter Index aiding faster retrieval than the normal un-indexed retrieval.

5. Indexing Algorithm

Part of Speech			
மோனை	MeterPattern1	Letter1	Words
		Letter2	Words
	MeterPattern2	Letter1	Words
		Letter2	Words
எதுகை	MeterPattern1	Letter1	Words
		Letter2	Words
	MeterPattern2	Letter1	Words
		Letter2	Words
இயைபு	MeterPattern1	Letter1	Words
		Letter2	Words
	MeterPattern2	Letter1	Words
		Letter2	Words

Figure 2. Indexing Logic

This indexing has been designed to facilitate fast retrieval of words specifically for lyric generation. For instance, the system would need a word of a particular meter pattern with a particular letter rhyming in monai scheme.

The data structure is organized as hash tables with separate tables for each Part Of Speech and Rhyming Scheme. For example, there is a separate table for Nouns' monai, Nouns' edhugai, Nouns' iyaibu and so on. Here, the keys are the Meter Pattern and the Letter at the particular Rhyming spot. The values are the words corresponding to the particular Meter pattern and letter.

The algorithm : The word database is scanned word by word and is indexed based on meter pattern and rhyme. Here,

$\alpha \leftarrow$ *number of words in the word database*;

$w_i \leftarrow$ *ith word in the word database*;

$\beta \leftarrow$ *meter pattern of* w_i ;

monaiKey $\leftarrow$ *key of wi corresponding to monai*;

edhugaiKey $\leftarrow$ *key of wi corresponding to edhugai*;

iyaybuKey $\leftarrow$ *lkey of wi corresponding to iyaybu*;

begin

$\forall_{i = 1, 2, 3 \ldots n}$ *do*

$\beta \leftarrow$ *MeterPattern of* w_i;

monaiKey $\leftarrow$ *firstLetter* (w_i) + β ;

edhugaiKey $\leftarrow$ *secondLetter* (w_i) + β;

iyaybuKey $\leftarrow$ *lastLetter* (w_i) + β ;

Add the word to the list of words with the respective keys in the respective Hashtables.

end;

For each word in the word database, meter pattern is extracted first followed by letters at the rhyming spots namely, first, second and last positions (for monai, edhugai and iyaibu respectively). Keys for monai, edhugai and iyaybu are found out using the abovesaid equations. The word is added to the monai, edhugai and iyaibu hashtables with keys monaiKey, edhugaiKey and iyaybuKey respectively if those keys don't appear previously in the tables. Else, the word is added to the list of words with that key.

Hash-tables are chosen for the implementation as they have the retrieval complexity of O(1).

6. Results

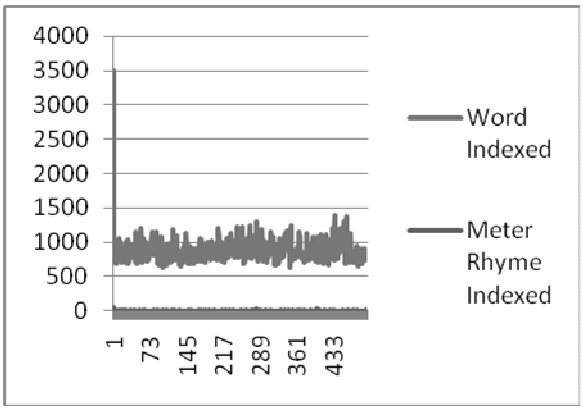


Figure 3. Word Indexed Vs Meter Rhyme Indexed Approach

Indexing has brought about a drastic increase in the speed of retrieval of the words rhyming with a given word. Using the Word Indexed approach, the time complexity was O(α) where α was the total number of words. But after Meter Rhyme indexing, the complexity has become O(1) due to the use of hash table which is a very efficient data structure for retrieval. Rhyming words for a sample of 500 words were retrieved using both the approaches and the above mentioned graph was obtained. The Word indexed system tooK 875.47millisecond in an average while the Meter Indexed system tooK 1.90millisecond only.From the graph it can also be inferred that there are steep variations in the Word-Indexed approach while the Meter-Rhyme Indexed approach does not show such steep variations and is consistent. In terms of time efficiency, Meter-Rhyme indexed approach is evidently superior compared to Word-Indexed approach. In terms of space, it is not so efficient as each word will occur not once, but nine times in various Hash tables.

7. References

- Hisar Maruli Manurung: "An evolutionary algorithm approach to poetry generation", Thesis for Doctor of Philosophy, University of Edinburgh, 2003.
- Sowmiya Dharmalingam., Madhan KarKy. "LaaLaLaa - A Tamil Lyric Analysis and Generation FrameworK" in World Classical Tamil Conference – June 2010, Coimbatore

- RamaKrishnan, A., S. Kuppan, and S.L. Devi. "Automatic Generation of Tamil Lyrics for Melodies" in NAACL HLT WorKshop on Computational Approaches to Linguistic Creativity. 2009. Colorado.
- K. Narayana Murthy "An Indexing Technique for Efficient Retrieval from Large Dictionaries", National Conference on Information Technology NCIT-97, 21-23 December 1997, Bhubaneswar.
- Eric Nichols, Dan Morris, Sumit Basu, Christopher Raphael, "Relationships between lyrics and melody in popular music", ISMIR 2009, October 2009, Japan.
- Anna Babarczy, IldiKó Bencze, István FeKete, Eszter Simon, "The Automatic Identification of Conceptual Metaphors in Hungarian Texts: A Corpus-Based Analysis", Proceedings of The seventh international conference on Language Resources and Evaluation (LREC), 2010, Malta.
- Xiao Hu, J. Stephen Downie, Andreas F. Ehmann, "Lyric Text Mining in Music Mood Classification", ISMIR 2009, Japan.
- Jose P. G. Mahedero, Alvaro Martınez, Pedro Cano, "Natural Language Processing of Lyrics", Proceedings of the 13th annual ACM internationalconference on Multimedia, New YorK, NY, USA, 2005.
- Rudolf Mayer, Robert Neumayer, Andreas Rauber, "Rhyme and style features for musical genre classification by lyrics", Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08), Philadelphia, PA, USA, September 14-18, 2008.
- Beth Logan, Andrew KositsKy, Pedro Moreno, "Semantic analysis of Song Lyrics", IEEE International Conference on Multimedia and Expo (ICME), June 2004.
- Yael Netzer, David Gabay,Yoav Goldberg, Michael Elhadad, "GaiKu : Generating HaiKu with Word Associations Norms", Workshop on Computational Approaches to Linguistic Creativity, CALC-2009 in conjunction with NAACL-HLT 2009, Boulder, Colorado.
- B. Logan, A. KositsKy, and P. Moreno, "Semantic Analysis of Song Lyrics", in Proc IEEE ICME, 2004.

Tamil Document Summarization Using Laten Dirichlet Allocation

N. Shreeya Sowmya[1], T. Mala[2]
[1]*Department of Computer Science and Engineering, Anna University*
[2]*Department of Information Science and Technology, Anna University*
Guindy, Chennai
[1]shreeya.mel@gmail.com [2]malanehru@annauniv.edu

Abstract

This paper proposes a summarization system for summarizing multiple tamil documents. This system utilizes a combination of statistical, semantic and heuristic methods to extract key sentences from multiple documents thereby eliminating redundancies, and maintaining the coherency of the selected sentences to generate the summary. In this paper, Latent Dirichlet Allocation (LDA) is used for topic modeling, which works on the idea of breaking down the collection of documents (i.e) clusters into topics; each cluster represented as a mixture of topics, has a probability distribution representing the importance of the topic for that cluster. The topics in turn are represented as a mixture of words, with a probability distribution representing the importance of the word for that topic. After redundancy elimination and sentence ordering, summary is generated in different perspectives based on the query.

***Keywords*- Latent Dirichlet Allocation, Topic modeling**

I. Introduction

As more and more information is available on the web, the retrieval of too many documents, especially news articles, becomes a big problem for users. Multi-document summarization system not only shortens the source texts, but presents information organized around the key aspects. In multi-document summarization system, the objective is to generate a summary from multiple documents for a given query. In this paper, summary is generated from the multi-documents for a given query in different perspectives. In order to generate a meaningful summary, sentences analysis, and relevance analysis are included. Sentence analysis includes tagging of each document with keywords, named-entity and date. Relevance analysis calculates the similarity between the query and the sentences in the document set. In this paper, topic modeling is done for the query topics by modifying the Latent Dirichlet Allocation and finally generating the summary in different perspectives

The rest of the paper is organized as follows. **Section 2** discusses with the literature survey and the related work in multi-document summarization. **Section 3** presents the overview of system design. **Section 4** lists out the modules along with the algorithm. **Section 5** shows the performance evaluation. **Section 6** is about the conclusion and future work.

II. Literature Survey

Summarization approaches can be broadly divided into extractive and abstractive. A commonly used approach namely extractive approach was statistics-based sentence extraction. Statistical and linguistic features used in sentence extraction include frequent keywords, title keywords, cue phrases, sentence position, sentence length, and so on [3]. Cohesive links such as lexical chain, co-reference and word co-occurrence are also used to extract internally linked sentences and thus increase the cohesion of the summaries [2, 3]. Though extractive approaches are easy to implement, the drawback is that the resulting summaries often contain redundancy and lack cohesion and coherence. Maximal Marginal Relevance (MMR) metric [4] was used to minimize the redundancy and maximize the diversity among the extracted text passages (i.e. phrases, sentences, segments, or paragraphs).

There are several approaches used for summarizing multiple news articles. The main approaches include sentence extraction, template-based information extraction, and identification of similarities and differences among documents. Fisher et al [6] have used a range of word distribution statistics as features for supervised approach. In [5], qLDA model is used to simultaneously model the documents and the query. And based on the modeling results, they proposed an affinity propagation to automatically identify the key sentences from documents.

III. System Design

The overall system architecture is shown in the Fig. 1. The inputs to the multi-document summarization system are multi-documents which are crawled based on the urls given and the output given by the system is a summary of multiple documents.

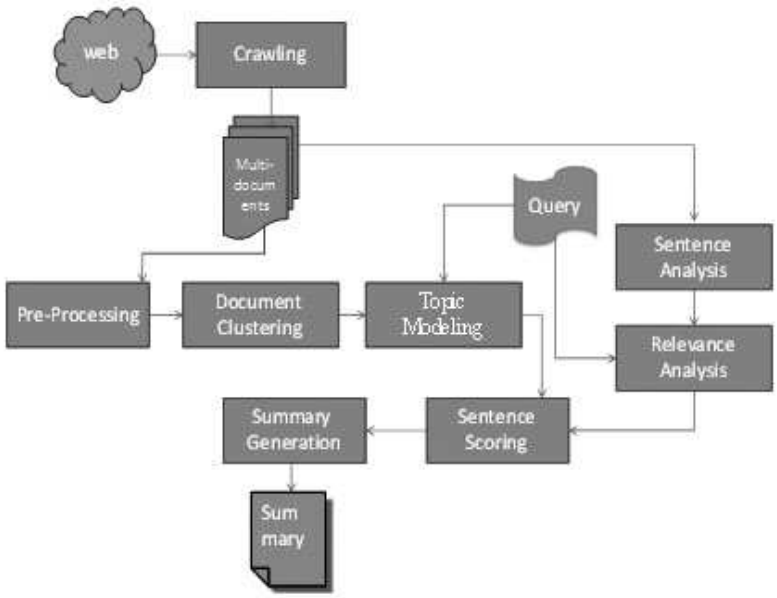


Fig. 1 System Overview

System Description

The description of each of the step is discussed in the following sections. The architecture of our system is as shown in Fig. 1.

1. Pre-processing

Pre-processing of documents involves removal of stop words and calculation of Term Frequency-Inverse Document Frequency. Each document is represented as feature vector, (ie.,) terms followed by the frequency. As shown in Fig. 1, the multi-documents are given as input for pre-processing, the documents are tokenized and the stop words are removed by having stop-word lists in a file. The relative importance of the word in the document is given by

$$Tfidf(w) = tf*(\log(N)/df(w)) \quad -(1)$$

where, tf(w) – Term frequency (no. of word occurrences in a document)

df(w) – Document frequency (no. of documents containing the word)

N – No. of all documents

2. Document clustering

The pre-processed documents are given as input for clustering. By applying the k-means algorithm, the documents are clustered for the given k-value, and the output is the cluster of documents containing the clusters like cricket, football, tennis, etc.., if the documents are taken from the sports domain.

3. Topic modeling

Topic models provide a simple way to analyze large volumes of unlabeled text. A "topic" consists of a cluster of words that frequently occur together. In this paper, Latent Dirichlet Allocation is used for discovering topics that occur in the document set. Basic Idea- Documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.

Sentence analysis

Multi-documents are split into sentences for analysis. It involves tagging of documents by extracting the keywords, named-entities and the date for each document. Summary generation in different perspectives can be done from the tagged document.

Query and Relevance analysis

The semantics of the query is found using Tamil Word Net. The relevant documents for the given query are retrieved. The relevance between the sentences and the query is calculated by measuring their similarity.

Query-oriented Topic modeling

In this paper, both topic modeling and entity modeling is combined [3]. Based on the query, the topic modeling is done by using Latent Dirichlet Allocation (LDA) algorithm. Query is given as prior to the LDA and hence topic modeling is done along with the query terms. Query may be topic or named-entity along with date i.e. certain period of time.

4. Sentence scoring

The relevant sentences are scored based on the topic modeling. For each cluster, the sum of the word's score on each topic is calculated, the sentence with the word/topic of high probability are scored higher. This is done by using the cluster-topic distribution and the topic-word distribution which is the result of the Latent Dirichlet Allocation.

5. Summary generation

Summary generation involves the following two steps

5.1 Redundancy elimination

The sentences which are redundant are eliminated by using Maximal Marginal Relevance (MMR) technique. The use of MMR model is to have high relevance of the summary to the document topic, while keeping redundancy in the summary low.

5.2 Sentence ranking and ordering

Sentence ranking is done based on the score from the results of topic modeling. Coherence of the summary is obtained by ordering the information in different documents. Ordering is done based on the temporal data i.e. by the document id and the order in which the sentences occur in the document set.

IV. Results

Table 1 shows the topic distribution with number of topics as 5, the distribution includes the word, count, probability and z value. The topic distribution is for each cluster.

Table 1 Topic model with number of topics as 5

TOPIC 0 (total count=1061)

WORD ID	WORD	COUNT	PROB	Z
645	அயோத்தி	42	0.038	5.7
2806	தீர்ப்பு	38	0.035	5.4
2134	இந்து	36	0.033	5.2
589	அமைப்பு	32	0.029	4.8
1417	அரசு	27	0.025	2.9
2371	லக்னோ	27	0.025	4.5
…				

V. Conclusion and Future Work

In this paper, a system is proposed to generate summary for a query from the multi-documents using Latent Dirichlet Allocation. The multi-documents are pre-processed, clustered using k-means

algorithm. Topic modeling is done by using Latent Dirichlet Allocation. The relevant sentences are retrieved according to the query, by finding the similarity between the sentences and the query. Sentences are scored based on the topic modeling. Redundancy removal is done using MMR approach.

Topic modeling can be extended to find the relationship between the entities, i.e. the topics associated with the entity as a future work.

References

- Arora.R and Ravindran.B, "Latent dirichlet allocation based multi-document summarization". In *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*, 2008, pp. 91–97.
- Azzam.S, Humphrey.K and Gaizauskas.R, "Using coreference chains for text summarization". In *Proceedings of the ACL Workshop on Coreference and its Applications*, 1999, pp. 77-84
- Barzilay.R and Elhadad.M, "Using lexical chains for text summarization". In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, 1997, pp.10-18.
- Carbonell.J, and Goldstein.J. "The use of MMR, diversity-based reranking for reordering documents and producing summaries". In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 24-28 August, Melbourne, Australia, 1998, pp. 335-336.
- Dewei Chen, Jie Tang, Limin Yao, Juanzi Li, and Lizhu Zhou. Query-Focused Summarization by Combining Topic Model and Affinity Propagation. In *Proceedings of Asia-Pacific Web Conference and Web-Age Information Management (APWEB-WAIM'09)*, 2009, pp. 174-185.
- Fisher. S and Roark.B, "Query-focused summarization by supervised sentence ranking and skewed word distributions," *Proceedings of the Document Understanding Workshop (DUC'06)*, New York, USA, 2006 pp.8–9.