Integration of Telugu dictionary into Tesseract OCR

MTech Stage II Project Report

Submitted in partial fulfillment of the

requirements for the degree of

Master of Technology

by

Manasa Gootla

123050093

under the guidance of

Prof. J. Saketha Nath

Dept of CSE, IIT Bombay

Dissertation Approval

This dissertation entitled **Integration of Telugu dictionary into Tesseract OCR**, submitted by **Manasa Gootla (Roll No:123050093)** is approved for the degree of **Master of Technology in Computer Science and Engineering** from **Indian Institute of Technology Bombay**.

[illegible]

Prof. Saketha Nath
Dept. of CSE, IIT Bombay
Supervisor

[illegible]

Prof. Sunita Sarawagi
Dept. of CSE, IIT Bombay
Examiner-I

[illegible]

Prof. Ganesh Ramakrishnan
Dept. of CSE, IIT Bombay
Examiner-II

[illegible]

Prof. Anirudha Joshi
IDC, IIT Bombay
Chairperson

Place: IIT Bombay, Mumbai
Date: 19th June, 2014

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

G. Manasa

Manasa Gootla

Roll No: 123050093

Date: 19th June, 2014

Place: IIT Bombay, Mumbai

Acknowledgements

I would sincerely like to thank my guide, Prof. J.Saketh Nath for his motivating support throughout the semester and the consistent directions that he has fed into my work. I would like to thank Prof. Ganesh Ramakrishnan for valuable guidance and discussions.

Contents

List of Tables

List of Figures

Abstract

Applications of OCR in digitizing printed and hand-written documents have found their way into preserving historical documents. OCR on Telugu language is particularly difficult because of the complexities in the script. The existing OCR systems for Telugu are TESSERACT and Drishti. An in-house OCR has been developed in an attempt to overcome the difficulties in recognizing Telugu characters. The comparision of performance of Tesseract and in-house system on a toy corpus is given. We implemented Myers algorithm to find and compare the accuracies of Tesseract and in-house system. A detailed insight into the Tesseract system and its novel approaches like line finding, page layout analysis, feature extraction, adaptive classification and an overview of the in-house system is also given. To improve the word accuracies of tesseract system, the dictionary has been populated with words, which are more likely to appear in the historical documents, along with their frequencies. The words that are currently in use, colloquial words, have also been added to the dictionary to make the OCR robust to the type of document given as input. We have also observed the effect of sandhis on the word recognition module.

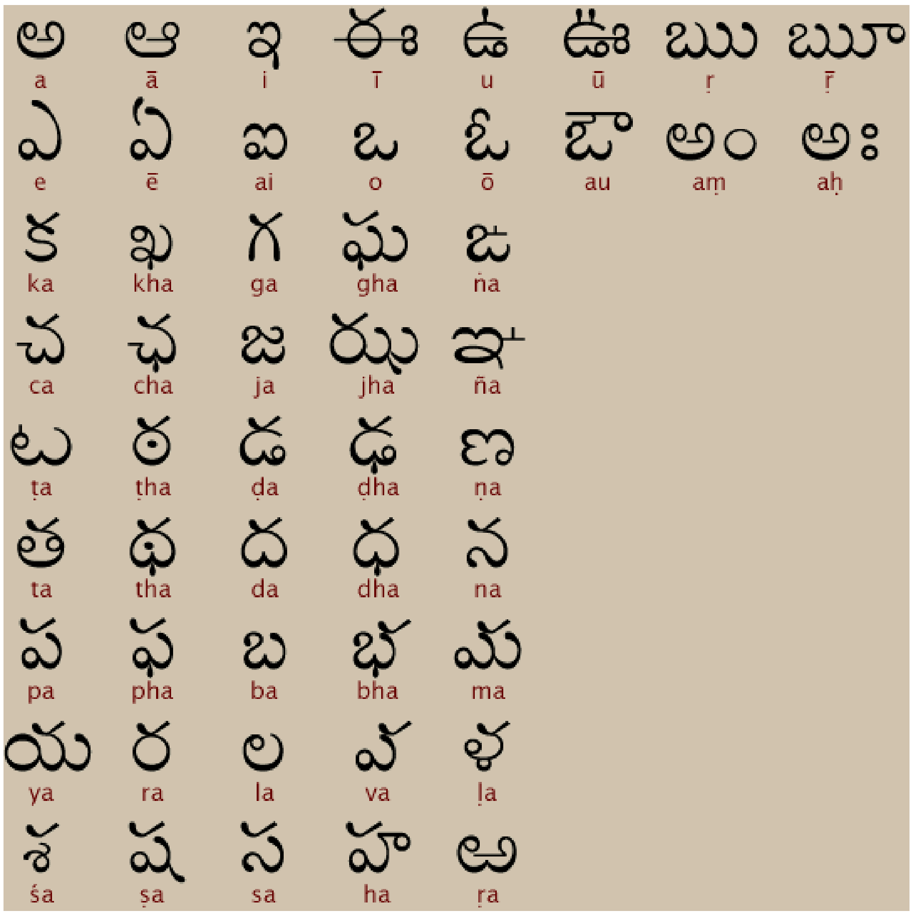
Chapter 1

Introduction

Optical Character Recognition(OCR) converts scanned images of text into machine-encoded format thereby enabling editing and searching data. This ability of OCR has led to huge research in this area in the past few decades. Significant amount of research has been done on OCR for languages like English, Chinese and Japanese. Its applications include: search through printed and handwritten documents, aid for visually impaired, check clearing for finance companies.

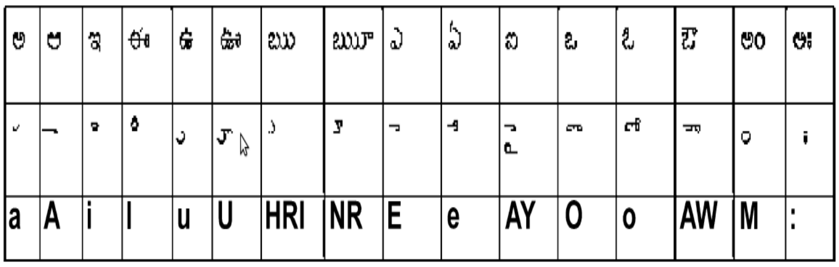
The existing OCR systems for English, Chinese cannot be adopted for Indian languages. Every character in English or Chinese are aligned in a single line and there aren't many discontinuities in the characters. OCR for Indian languages is much more difficult because of the Script(lipi) as more strokes are involved. A compound character(*Samyuktas*) can be formed from a huge combination of characters imposing complexities. Active Research has been done in recent years on Devanagari, Hindi, Kannada, Tamil and Telugu characters.

Telugu script is comprised of 16 vowels and 36 consonants(Figure 1.1). A compound character can be formed from a consonant and a vowel or it can be formed from 2 consonants and a single vowel leading to possible combinations of 36x16 and 36x36x16 respectively. When compared to other Indian languages, it has additional complexity involved beacause of the similarities in between many of the characters in the script. The nuances in these characters are difficult to catch even by human readers posing much more challenge for the OCR(Figure 1.3 and Figure 1.4). It is clear from Figure 1.4 that the compound character has many disjoint connected ocmponents associated with it.



అ a	ఆ ā	ఇ i	ఈ ī	ఉ u	ఊ ū	ఋ ṛ	ౠ ṝ
ఎ e	ఏ ē	ఐ ai	ఒ o	ఓ ō	ఔ au	అం aṃ	అః aḥ
క ka	ఖ kha	గ ga	ఘ gha	ఙ ṅa			
చ ca	ఛ cha	జ ja	ఝ jha	ఞ ña			
ట ṭa	ఠ ṭha	డ ḍa	ఢ ḍha	ణ ṇa			
త ta	థ tha	ద da	ధ dha	న na			
ప pa	ఫ pha	బ ba	భ bha	మ ma			
య ya	ర ra	ల la	వ va	ళ ḷa			
శ śa	ష ṣa	స sa	హ ha	ఱ ṟa			

Figure 1.1: Alphabets in Telugu script



అ	ఆ	ఇ	ఈ	ఉ	ఊ	ఋ	ౠ	ఎ	ఏ	ఐ	ఒ	ఓ	ఔ	అం	అః
✓	ా	ి	ీ	ు	ూ	ృ	ౄ	ె	ే	ై	ొ	ో	ౌ	ం	ః
a	A	i	I	u	U	HRI	NR	E	e	AY	O	o	AW	M	:

Figure 1.2: Vowels and their associated Modifiers

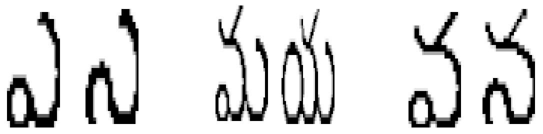


Figure 1.3: Confusing characters

1.1 Motivation

The existing OCR systems for Telugu are TESSERACT[7] and Drishti [1] have constraints on images like quality, resolution, font, input/output formats. Inspite of these constraints

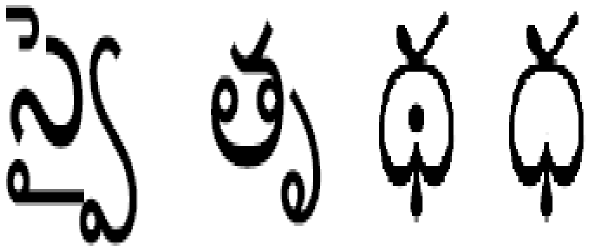


Figure 1.4: Compound characters and complexities

the accuracy levels achieved by these systems are not satisfactory, the best of these being Tesseract. Tesseract has only considered Pothana fonts making it vulnerable for font independent system. Drishti showed better results only when the scans are of laser quality prints. An in-house system has been developed that helps in overcoming these constraints [5] and [6].

Chapter 2

Stage1 Summary

In Stage1, we have studied the implementation of Tesseract software in detail and contrasted it against the in-house system. Each OCR has primarily 3 stages : preprocessing, Feature extraction and Classification. In the following sections we give a brief insight into the differences of the two OCR systems in discussion in each stage.

2.1 Comparision

In Tesseract, the image is stored in a raw format in the data structure IMAGE. The leptonica library uses pix data structure to refer to this raw image. Instead of messing with the raw data, all the image processing operations are done on this pix data structure. Image is stored as data structure Pix (Leptonica style). The image is scanned row wise and the rgb info is stored in a color map. The width, depth, height, the raster data and the number of pix pointers referring the particular pix object are stored.

In in-house system, Java imaging API stores the input as BufferedImage, which is an Image with an accessible data buffer. It is therefore more efficient to work directly with BufferedImage. A BufferedImage has a ColorModel and a Raster of image data. The ColorModel provides a color interpretation of the image's pixel data.The Raster performs the following functions:
-Represents the rectangular coordinates of the image
-Maintains image data in memory
-Provides a mechanism for creating multiple subimages from a single image data buffer

-Provides methods for accessing specific pixels within the image

2.1.1 Pre-processing

In pre-preocessing, the first step is image binarization. The image is binarized so that the text is in black representing foreground and the rest is in white representing the background. Tesseract used a global binarization technique called Otsus thresholding method whereas the inhouse system used MaxVariance threshold (Java Imaging API) for the global binarization.

The next step is character and word segmentation, in which connected components are found through 8-way flood fill algorithm. In Tesseract, auto page segmentation is done initially to mask the photos in the image and also page is divided into multiple blocks if there are multiple columns. For each block, outlines are extracted from the connected components through edge detection, which are then gathered and nested into blobs which give the average character spacing. The words are then split through character spacing.

Tesseract also does skew detection and correction by tracing along the bounding boxes of the blobs. The page layout analysis provides text regions of roughly uniform text size. Drop caps are removed by simple percentile filter. Small blobs like punctuation, noise are removed by taking threshold a fraction of median height. The characteristics of the font are determined by x-height, ascender height, fixed pitch, baseline, median line, descender height which are calculated from page layout analysis. The filtered blobs are sorted by x-coordinates so that they are uniquely associated with text line. The slope is tracked across the page using the baseline. Once the filtered blobs are assigned to their respective text lines, the filtered out blobs are fitted back into appropriate lines.

The in-house system uses Hidden Markov Model to model the characteristic of each telugu line i.e., white pixels, followed by black pixels with low density, black pixels with high density and black pixels with low density. The pattern is fed to the 4-state HMM with density of black pixels as feature. SVM Torch API has been used for HMM implementation. In vertical segmentation, since each word has significant gap between its neighbours when compared to gaps within word, each segmented line is processed for significant white pixel density and thus words are segmented[3].

2.1.2 Feature Extraction

In training process, a set of images consisting of characters are gathered. For frequent characters, a minimum 30 samples per each character, for moderate characters 20-30 samples and for less frequent 5-10 samples are to be provided for training. Manually for each character, a bounding box is to be created coordinates and the corresponding ASCII code. The vowel modifiers can be trained seperately too with the bounding box defined only for those. In tesseract, the features extracted in training data are the segments of polygonal approximation. For recognition i.e., for unknown characters, features of a small, fixed length are extracted from outline and are matched many to one against the clustered prototype features of the training data. The features for prototypes are 4-dimensional (x,y position, angle, length) with 10-20 features in a prototype configuration. The features for unknown are 3-dimensional (x, y position, angle) and for each character there are approximately 50-100 features. The normalized features for the unknown are computed by tracing around the outline of the blob unitwise. the x,y position and the angle between the tangent line and a vector eastward from the centre of the blob are saved as feature.

In the in-house system, wavelet features have been extracted by dowsampling the character image to an approximation image and three images: vertical, horizontal and diagonal. By downsampling, the directional features of the image are captured which is Battle-Lemarie filter is used as wavelet basis function to capture the circular property of Telugu characters. Other features were also experimented on i.e., Gabor features, skeleton features, circular zonal features, out of which wavelet features were found to give better results.

2.1.3 Classification

In Tesseract, classfication proceeds as a two step process: in the first step: each feature in the unknown matched against 3-D look-up table and fetches a bit vector of all the classes it might match. The bit vectors are summed over all features. The classes with the highest counts become shortlisted for the next step of classification. This step is called class pruning. In the second step: each feature of unknown looks up for bit vector of prototypes of given class it matches. Each prototype character class is logical sum of

product expression where each term is configuration. So the distance is evaluated and the total similarity evidence of each feature in each configuration as well as each prototype is recorded. The best combined distance which is calculated from summed feature and prototype evidence is the best over all stored configuration of the class. The data structure used is a k-d tree (k- dimensional tree) for each character. Nearest negihbour classification is employed and the best choice is returned during the traversal of the tree. Each character classification is associated with two numbers:
- confidence : prototype - normalized distance
- rating : confidence * (total outline length of the character)

In in-house system, the feature vectors generated are passed onto the classification stage. The unicode of the letters thus classified are generated individually for the top , middle and bottom parts of each character. The unicode of all these are combined to generate the compound character.

2.2 Adaptive Binarization

As a preprocessing task, binarization affected the later stages thereby effecting the OCR accuracies. This is because of the global binarization technique where the threshold value is chosen from the whole image. Several local binarization methods[4] were tested to capture the characteristics of the historical documents viz, no proper illumination and fading. Adaptive binarization using Sauvolas technique is found to give better binarization result. There was an increase of 4-5% on an average in the character accuracies per document.

2.3 Proposed improvement

Tesseract has word recognition module which has dictionary files to improve the word accuracies. Tesseract for telugu does not implement the word recognition module. In the next chapter, we discuss the creation of dictionary files and the problems faced in collecting the required data and how they effected the overall character and word accuracies.

Chapter 3

Word Recognition

Linguistic module (Permuter) contains language files freq dawg, word dawg, user words, unicharset, DangAmbigs. DAWG (Directed Acyclic Word Graph) file is a datastructure that facilitates fast search for words. The dictionary can be stored in the word dawg file. This cannot be changed by the user. The user can modify user words dawg file and personalize the word recognition module. The start node is the first character for the current word to be searched. These files are not defined for Telugu language resulting in much lower accuracies than that of English, chinese. These files ensures the correct word inspite of few characters being wrongly classified in the word.

Permuter chooses best word string from each of the following categories:top frequency word, top dictionary word, top numeric word, top upper case, top lower case, top classifier choice The final decision for a given segmentation is the word with the lowest distance rating. The classifier stores the top best choices for each character during the character recognition process.

Words from different segmentations may have different numbers of characters in them. It is hard to compare these words directly, even where a classifier claims to be producing probabilities, which Tesseract does not. This problem is solved in Tesseract by generating two numbers for each character classification. The first, called the confidence, is minus the normalized distance from the prototype. This enables it to be a confidence in the sense that greater numbers are better, but still a distance, as, the farther from zero, the greater the distance. The second output, called the rating, multiplies the normalized distance from the prototype by the total outline length in the unknown character. Ratings for

characters within a word can be summed meaningfully, since the total outline length for all characters within a word is always the same. The word confidence is calculated as:
WERD_CHOICE* choice = word->best_choice
int w_conf = 100 + 5 * choice->certainty()

3.1 Chopping joined characters

While the result from a word is unsatisfactory, Tesseract attempts to improve the result by chopping the blob with worst confidence from the character classifier. Candidate chop points are found from concave vertices of a polygonal approximation of the outline, and may have either another concave vertex opposite, or a line segment. It may take up to 3 pairs of chop points to successfully separate joined characters from the ASCII set. Chops are executed in priority order. Any chop that fails to improve the confidence of the result is undone, but not completely discarded so that the chop can be re-used later by the associator if needed.

3.2 Associating chopped characters

When the potential chops have been exhausted, if the word is still not good enough, it is given to the associator. The associator makes an A* (best first) search of the segmentation graph of possible combinations of the maximally chopped blobs into candidate characters. It does this without actually building the segmentation graph, but instead maintains a hash table of visited states. The A* search proceeds by pulling candidate new states from a priority queue and evaluating them by classifying unclassified combinations of fragments. It may be argued that this fully-chop-then-associate approach is at best inefficient, at worst liable to miss important chops, and that may well be the case. The advantage is that the chop-then-associate scheme simplifies the data structures that would be required to maintain the full segmentation graph.

Chapter 4

Experiments

We have added word data file and frequent word data files for Telugu, as they were not not officially available in the current version of Tesseract. The word data file is a dictionary that enlists the words in alphabetical order. The frequent word list contains the most frequently occuring words in Telugu.

We have observed that the addition of word data files effected the character accuracies because of the chopping and association being done in the word recognition module as mentioned in the earlier section. Wrongly classfied characters are replaced by the correct characters as per the words in the dictionary if the correct character to be replaced is available in the top best choices of the character. If it is not available, then the character is replaced by a character that is most likely to be the character to be appearing in the dictionary during the traversal from the letter preceding it. Some characters which were correctly recognized were now wrongly classified because if the word confidence is not satisfactory, the chopping and then the A* graph during association look for the next best character replacement that also can be found in the dictionary file during the traversal in the dawg. We have experimented with dictionaries that contain only colloquial words , only historical words, a unified dictionary.

We have also populated freq word list with the most frequent words. We have experimented on the number of words the freq list should have without compromising on the accuracies. The way it works is that words in the freq-words list are weighted higher when Tesseract looks for likely matches. So if there are too many, it is likely that we get less common variants switched to the most common ones. We have also experimented with dictionary consisting of only limited historical words(300) tightly bound to the text in toy corpus and then with dictionary consisting of only colloquial words(3M) and then with historical words unrelated to the toy corpus. We have observed that many characters are being confused with others. We have also observed that since the vowel modifiers were included in the training, the output resulted many characters broken into consonant and vowel modifier. Though the characters were recognised correct, the output was mostly filled with vowel modifiers seperated from the consonants.

Chapter 5

Results

Implemented Myer's algorithm[2] to find the differences between the actual text of the image and the output text produced by the OCR system. The algorithm was implemented in python. The output is returned as a list where all the characters that are inserted are preceded by a '+1' and that are deleted by a '-1' and those characters which remain same are preceded by a '0'. The accuracy is calculated as:

$$A = \frac{r}{c}$$

where r = the number of characters that are correctly recognized c = number of characters in the actual text

The character and word level accuracies for Tesseract system without and with our integrated binarization module and Telugu specific dictionary files are tabulated in table 5.1. We can see a significant increase in the accuracy levels. The addition of dictionary files also effected the character accuracies, because of the chopping and association of characters to get the most probable word that could be in the dictionary.

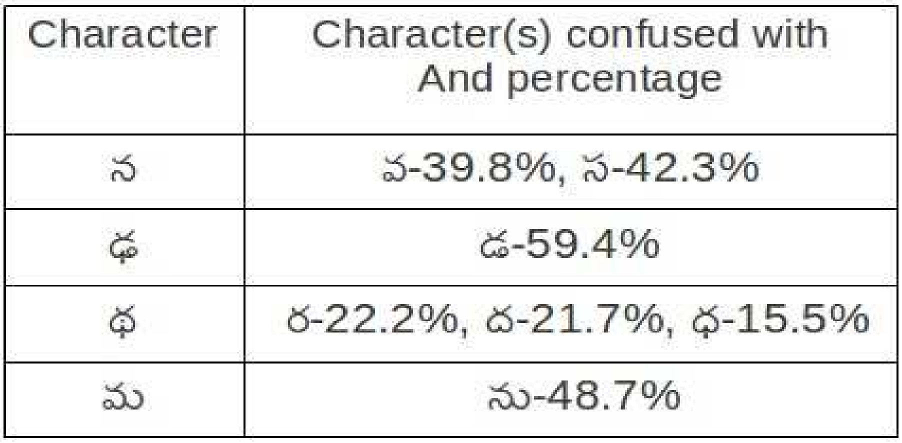
We have experimented on the recent documents with the changed dictionary files. The results are tabulated in table 5.2.

We have also observed that many characters are being confused with others. The top 5 characters which are observed to be more confused are tabulated in figure 5.1

For the tables below the first row indicates character accuracies and the next row indicates word accuracies.

Table 5.1: Comparison of accuracies for historical documents

Filename	Tesseract OCR	with binarization and dictionary files
fwdpayasam/scan0002.jpg	58.384% 15%	65.75% 20%
fwdpayasam/scan0003.jpg	66.54% 20%	72.83% 23%
oct01/govu1/scan0006.jpg	68.95% 34%	78.92% 38%
oct01/govu1/scan0007.jpg	77.03% 30%	79.57% 33%
oct01/govu1/scan0008.jpg	77.38% 32%	79.54% 34%
oct01/ks4/scan0102.jpg	71.36% 10%	74.07% 23%
oct01/govu5/scan0082.jpg	38.7% 19%	78.12% 29%
oct01/govu5/scan0083.jpg	71.28% 22%	77.91% 28%



Character	Character(s) confused with And percentage
న	వ-39.8%, స-42.3%
ఢ	డ-59.4%
థ	ర-22.2%, ద-21.7%, ధ-15.5%
మ	ను-48.7%

Figure 5.1: Top confused characters

Table 5.2: Comparison of accuracies for the modern documents

Filename	Tesseract OCR	with binarization and dictionary files
Doc01	37.29% 16%	74.63% 24%
Doc02	17.7% 21%	35.83% 26%
Doc03	55.4% 4%	56.19% 9%
Doc04	56.34% 3%	57.62% 8%
Doc05	62.4% 11%	64.06% 20%
Doc06	65.26% 11%	66.05% 15%
Doc07	67.48% 11%	68.19% 18%

Chapter 6

Conclusion and Future Scope

We have observed that the dictionary can never be completely populated with all the words because of the complex morphology of the language. The union of two words results in a morphology where the last character of first word and the first character of second word are combined to get another character and the rest of the characters remain the same. Sandhis and samasas result in countless combination of words which cannot be included in the dictionary unless we use a sandhi splitter making the recognition of words much simpler. This phenomena results in an infinite words list which makes the purpose of dictionary useless. This problem can be overcome by using a sandhi splitter, which will break a word into its root words which can be easily found in the dictionary of words.

The quality of the image has a lot of impact in the later stages in the process of character recognition. We have observed that during the binarization process, the characters are broken when theres a thin line and faded background and if the characters which are at the back cover are visible to the front cover and the background is light, then these characters are also included on the front page which leads in garbled image. This can be overcome by focusing more on the digital processing of the image.

Training the character variants is highly difficult due to the huge number of combination of characters possible and the nuances add to the complexity. One possible solution is the brute force method of training all variants of characters with large number of samples per variants instead of training the vowel modifiers alone. The other approach is to instead the output can be processed for finding the disconnected character variants and joining them.

Bibliography

[1] Web Refererence http://www.ildc.in/Telugu/htm/lin_ocr_spell.htm. Drishti telugu ocr.

[2] EUGENE W. MYERS. An o(nd) difference algorithm and its variations. *Algorithmica*, pages 251–266, 1986.

[3] Srikanta Pal Nallapareddy Priyanka and Ranju Manda. Article:line and word segmentation approach for printed documents. *IJCA,Special Issue on RTIPPR*, pages 30–36, 2010.

[4] B. Gatos K. Ntirogiannis I. Pratikakis. Dibco 2009: document image binarization contest. *IJDAR*, pages 35–44, 2011.

[5] J. Saketha Nath Rajesh Arja. Ocr for historical telugu documents. *MTP Stage 1*, 2011.

[6] J. Saketha Nath Rajesh Arja. Ocr for historical telugu documents. *MTP Stage 2*, 2012.

[7] Ray Smith. An overview of tesseract engine.

Appendix

The following section provides the results of the experimentation with dictionary consisting of only limited historical words(300) tightly bound to the text in toy corpus and the next column with dictionary consisting of only colloquial words and the next column of colloquial words along with historical words unrelated to the toy corpus

Filename	Colloquial words	Historical	Unrelated words
fwdpayasam/scan0002.jpg	63.15% 15%	65.07% 17%	51.63% 9%
fwdpayasam/scan0003.jpg	66.54% 20%	72.83% 23%	61.5% 15%
oct01/govu1/scan0006.jpg	76.21% 25%	76.12% 26%	65.45% 21%
oct01/govu1/scan0007.jpg	78.7% 24%	78.82% 26%	78.41% 21%
oct01/govu1/scan0008.jpg	78.12% 25%	78.01% 28%	75.8% 20%
oct01/govu5/scan0082.jpg	43.04% 16%	43.04% 18%	42.9% 13%
oct01/govu5/scan0083.jpg	55.48% 15%	55.25% 16%	61.25 18%