CHAPTER 2: LITERATURE REVIEW

2.1. LITERATURE REVIEW

Wan-Yu Lin [31] introduced a framework that identifies online plagiarism by exploiting lexical, syntactic and semantic features that includes duplication-gram, reordering and alignment of words, POS and phrase tags, and semantic similarity of sentences. They establish an ensemble framework to combine the predictions of each model. Results demonstrate that their system can not only find considerable amount of real-world online plagiarism cases but also outperforms several state-of-the-art algorithms and commercial software.

Efstathios Stamatatos [32] presented a novel method for detecting plagiarized passages in document collections. In contrast to previous work in this field that uses mainly content terms to represent documents, he proposed a method based on structural information provided by occurrences of a small list of stopwords. The research elaborated that stopword n-grams are able to capture local syntactic similarities between suspicious and original documents. Moreover, an algorithm for detecting the exact boundaries of plagiarized and source passages was proposed.

DANIEL R. WHITE and MIKE S. JOY [33] presented algorithm for comparison of suspect documents at a sentence level and have implemented it as a component of plagiarism detection software for detecting similarities in both natural language documents and comments within program source-code. The algorithm is capable of detecting sophisticated obfuscation (such as paraphrasing, reordering, merging, and splitting sentences) as well as direct copying.

Finkel et al. [34] presented algorithm based on signature extraction. A signature of registered text is stored permanently and compared against other stored signatures. They compared various approaches based on storage of extracts and entire document. SCAM requires 30-60% of space for storage of extracts whereas SE requires approximately 5% of
size of documents. As compared to Signature Extraction, an Overlapping Chunks method is very accurate but it is far more expensive. Sqrt method and variance methods are used for culling. It is found that sqrt method does not store enough chunks for short files to allow reasonable testing whereas variance method does better.

Schleimer et al. [35] introduced local document fingerprinting algorithm, which captures an essential property of any fingerprinting technique guaranteed to detect copies. They have also presented winnowing, an efficient local fingerprinting algorithm, and have shown that winnowing’s performance is within 33% of the lower bound for fingerprinting algorithms.

Sebastian Niezgoda and Thomas P. Way [36] have presented SNITCH: A Software Tool for detecting Cut and Paste Plagiarism that used Google web API. They also discussed issues related to plagiarism detection software and empirical results along with performance and accuracy study were also presented.

Leonardo Mariani and Daniela Micucci [37] developed AuDeNTES – a tool that detects plagiarism in student submitted codes via the code fragments. The code fragments better represent the individual students’ contributions by filtering from students’ submissions the parts that might be common to many students due to the suggestions in the text of the exam. The filtered parts are identified by comparing students’ submissions against a reference solution, which is a solution of the exam, developed by the teachers.

Vishal Gupta, G.S. Lehal [38] have discussed the challenges in the pre-processing phase of Punjabi text summarization. They have designed a Punjabi word stemmer, stop word remover and have done significant work on Punjabi sentence boundary identification and identification of Punjabi cue phrase in a sentence.

Mandeep Singh Gill et. al. [39] have proposed a “Part-of-Speech Tagger” for Grammar Checking of Punjabi text. They investigated the use of a new tag set device for the Punjabi language and presented ground-breaking innovations in parts-of-speech tagging. The authors recommended this for grammar checking of not only Punjabi but also other languages of the Modern Indo-Aryan family.
Vishal Gupta, Gurpreet Singh Lehal [40] have done significant work on features selection and weight learning for Punjabi Text Summarization. The importance of sentences is decided based on statistical and linguistic features of sentences. For Punjabi language text Summarization, some of statistical features that often increase the candidacy of a sentence for inclusion in summary are: Sentence length feature, Punjabi Keywords selection feature (TF-ISF approach) and number feature. They have used Mathematical regression to estimate the text feature weights based on fuzzy scores of sentences of Punjabi news documents.

Imtiaz Hussain Khan et. al. [41] have proposed a dual architecture for detecting plagiarism in Arabic text documents. A dual component based system is proposed with local component searching for important terms in the query document and the global component using the output from local component to query Google search engine using Google APIs to find the matching documents from the internet domain.

Salha M. Alzahrani et. al. [42] have differentiated between literal and intelligent plagiarism. A detailed taxonomy of plagiarism has been discussed. A comparison of various similarity detection measures along with their comparison has also been presented in the paper. The authors assert that existing plagiarism detection tools only focus on copying of text, but are unable to detect intelligent plagiarism when the same idea is presented in different words.

Manber [43] presented a tool called sif for finding similar files in a depository of large file system. In this approach, the files are considered to be similar if they have significant number of common chunks. The running time complexity of the system is of order of 500MB to 1GB per hour. Sif uses the finger print approach to find the similarity of the documents and it works in two modes, one against all and all against all. The approach used in sif is completely syntactic and not context information has been included in it. This means the text in two documents which results in same meaning but the words are different are not recognized by this method.
Baker [44] created a program called “dup” which is capable to locate the instances of duplication of source code in the software. For further analyses of the matches, post processing is also done to improve the results. Dup is designed for C language which finds parameterized matches by an algorithm based on parameterized suffix trees. Dup is quite fast as it takes around 7.2 minutes plus 7 seconds for post processing to process almost 11M lines with a threshold of 30 lines. The measurement of time is taken on one 40MHz R3000 processor (primary I and D cache 64KB, secondary lMB, main 256MB, SGI IRIX 4.1). The duplication increases manifolds if the match length becomes small. Therefore, threshold lengths of 20 or above has been used in dup.

Shivakumar and Gracia-Molina [45] developed SCAM (Stanford Copy Analysis Mechanism) to improve COPS. Word frequency of documents is used to detect the duplicacy of the text. The mechanism employs exact string matching only. The system detects partial overlaps but comparison of those documents misleads which share many words. Researchers compared main memory requirement, disk storage requirement, and response time for registration, for querying. The system is tested on 50,000 internet news articles. It results in many false positives on two documents having similar word distribution but different in subject. The results are improved by excluding the most common words, and by shifting the mechanism to use the n-grams for accuracy.

Monostori et al. [46] introduced a system called MatchDetectReveal(MDR). The system is capable of identifying overlapping of text in documents and hence to report amount of plagiarism. The main component of MDR is matching engine which uses modified suffix tree representation. The modified suffix tree uses modified Ukkonen’s algorithm and hence is capable to identify the exact overlapping chunks. The running time of the system is compared for algorithm run on single machine and the algorithm run on multiple machines. It is found that the speed up is linear in first case. This is because nodes get congested when multiple nodes are used. Authors claim to develop various enhancing components in futures. Some of the components to be developed in future are like visualizer for user interface, search engine for identifying candidate documents to be compared, document generator for performance analysis, similarity and rule interpreter component for identifying inexact matches.
Lyon et al. [47] used the statistical method for fingerprinting text. The text is divided into the trigrams and the trigrams of two documents in picture are compared. The similarity measure is taken from Broder work. The system is incorporated with some preprocessing steps like decapitalization and punctuation removal. The system uses the corpus of 335 news reports. The identifying fingerprint associated with text is extracted from word trigrams. Some pre-processing techniques such as decapitalization, removing punctuation etc. is also done. The system has been developed with the corpus of 335 TV news reports. The system claims to identify similar as well as identical documents.

Collberg et al. [48] introduced a method detecting the self plagiarism and named it SPLAT. The system uses a WebL web spider that crawls through the web sites. The websites that are crawled are the top fifty Computer Science departments. The spider downloads various research papers from the websites and group them by author name. A text comparison algorithm is used to search for instances of textual reuse. The algorithm works at the sentence level and check similarity of documents on the basis of both identical and derived sentences. Instances of potential self plagiarism for each author are reported in an HTML document.

Frakes and Baeza-Yates [12], suggested a brute force approach for word stemming that uses a table lookup method for stemming. A list of all possible stems and their inflections is prepared. While checking for a possible stem, a brute force is performed on this list to find the root word. This approach provides accurate results for the words which are already present in the lookup table, and fails to stem the unknown words. Moreover the approach is language dependent.

Hafer and Weiss [49], suggested a word stemming approach called “Successor Variety Stemmer” that identifies the morpheme boundaries based on the available lexicon and decides where the words should be broken to get a stem. The approach is language independent. Authors suggest that the successor variety sharply increases when a segment boundary is reached by adding more amount of text thereby decreasing the variety of substrings. This information is used for finding the stem.
Lovins [50] defined the suffix stripping algorithm that uses the longest match first suffix stripping for finding the root word. A set of rules is defined with common possible suffixes. The word to be stemmed is compared against the list and the matching suffixes are dropped from the word resulting in the root word. Lovins also suggested the recursive procedure to remove each order-class suffix from the root word.

Martin Porter [51] suggested an improved method for suffix stripping which was less complex as compared to Lovin’s method. However the improvement over the Lovin’s results was not much significant. Both these algorithms are best suitable for less inflectional language like English, and their extension to other languages required the entire rule set to be re-defined as per the language.

Majumdar et. all [52] suggested YASS: Yet Another Suffix Stripper that uses a clustering-based approach to discover equivalence classes of root words and their morphological variants. A set of string distance measures are defined, and the lexicon for a given text collection is clustered using the distance measures to identify these equivalence classes. Its performance is comparable to that of Porter’s and Lovin’s stemmers, both in terms of average precision and the total number of relevant documents retrieved.

Sajjad Khan et. all [53] have discussed a Template Based Affix Stemmer for a Morphologically Rich Language, that not only depends on removing prefixes and suffixes but also on removing infixes from Arabic text.

Gupta Vishal, Lehal Gurpreet [54] developed a rule based Punjabi word stemmer, that could stem nouns and proper names only.

2.2. EXISTING PLAGIARISM DETECTION TOOLS

2.2.1. Turnitin

The Turnitin.com provides an industry leading plagiarism detection engine. The tool is able to detect plagiarism in the query documents by comparing it with the various internet
sources, research papers and student reports. The user needs to submit the documents to the online portal and is provided with different options like selecting the size of minimum matches, specifying the exclusions from the comparisons and exclusion of bibliographic text. The tool when posed with a multi-lingual document containing English, Hindi and Punjabi text failed to provide the accurate results (Figure 5 & Figure 6). For testing purpose, a sample research paper consisting of Hindi and Punjabi and English was downloaded from Internet and submitted to the Turnitin platform. Logically, this paper was supposed to give 100% plagiarism as the paper itself was downloaded from Internet, However, it calculated only 72% similarity. On further investigations, it was found that the Turnitin engine only detected the English words, but failed to recognize the Punjabi and Hindi text at all.

Figure 5. Turnitin Report showing in-correct similarity score
Figure 6. Incorrect processing of Hindi and Punjabi Text

When posed with a mono-lingual Punjabi document containing Punjabi news article from a popular online newspaper, the tool altogether failed to recognize the Unicode Punjabi text and calculated a similarity score zero (Figure 7, Figure 8).

Figure 7. Turnitin Report for Mono-lingual Punjabi document in Unicode format
2.2.2. Urkund

Urkund provides online plagiarism detection services to its subscribers. The users need to create an account with Urkund for getting their research work checked against the plagiarism. Many universities and government agencies have tie-ups established Urkund for checking plagiarism in the documents submitted to them. The Urkund tool also was tested with exact copy of a newspaper article as well as a re-phrased copy of the article. Since Urkund claims to support Unicode encoding schemes, the tool can work with Hindi and Punjabi text apart from the regular English text. However, on further investigations, it was found that the tool could provide only the basic functionality for the Punjabi document (i.e. string comparison only) as it detected the 100% match with the original article written on the source website, but failed to detect the plagiarism in the re-phrased document (Figure 9, Figure 10, Figure 11).
Figure 9. Urkund Plagiarism report of Original, Unchanged document

Figure 10. Urkund Plagiarism report of re-phrased document
2.2.3. WCopyFind

WCopyFind tool is a standalone tool available for downloading on the user’s machines. The software is available under Creative Commons License. After simple installation on users’ machine, the software can run standalone, from where the plagiarism can be checked. The WCopyfind tool since performs comparison on local files only, it’s use is restricted on the availability of the comparison documents. The software also provides a number of configuration options for plagiarism detection, such as including or excluding punctuation marks, ignoring numbers, match cases, skip longer words and minimum % of matching words etc. WCopyfind supports many foreign languages, but unfortunately, the Punjabi language is not supported by the tool (Figure 12 & Figure 13). Although the comparison on local systems is done in Unicode format, however the final report is not Unicode compatible and hence the corresponding matches from the different documents were not indicated.
The user can also specify the size of the shortest phrase (nGram) match. In case of large documents, the output can be restricted to specified number of words only. The system also allows to configure the number of most imperfections to be allowed in the final result.

The output of WCopyFind is stored in the specified folder in the form of HTML documents. WCopyFind also provides option to build vocabulary for the various languages. In the Comparison result, the matches from the various files are indicated in the list, specifying filename, %age of perfect match as well as %age of Overall match.. The users can double click to get details of each match from the specific document.
2.2.4. PlagScan

PlagScan is a web based online plagiarism detection tool. The website allows free registration to the users and provides a free plagiarism assessment to single article of maximum 2000 words. The tool when posed with Punjabi text in Unicode format failed to detect the text as it was in Unicode format. The figures below show the results –
2.2.5. Small SEO Tools Plagiarism Detection

The website smallseotools.com provides a plagiarism detection tool that can detect plagiarism in documents of size upto 1000 words. The website compares the given text
with various internet sources. The result is displayed in two formats – online as well as offline in the form of pdf export. The tool failed to detect the plagiarism in Punjabi text. The online output displayed the Punjabi text, however the PDF report failed to correctly render the Punjabi text. The figures below show the sample outputs from online as well as PDF formats.

Figure 17. SEOTools.COM online plagiarism checker report
2.2.6. PlagiarismSoftware.net

The website PlagiarismSoftware.Net provides online plagiarism checking tool. The tool compares the given text to over billions of web pages indexed using their proprietary...
plagiarism software. It provides a sentence based comparison. The results in detail are shown online. The website accepts Microsoft word or txt file to check content for uniqueness. The tool can also check any web page content by just providing the URL. The website supports Unicode text, but failed to detect plagiarism in the documents with altered text.

The figures below show the sample output using the tool.

![Figure 19. Plagiarism Checker by PlagiarismSoftware.Net](image)

### 2.2.7. QueText

QueText Provides very basic plagiarism detection tool that can predict similarity score between the suspected matches. The tool however could not provide any comparison to the Punjabi text copied from some popular newspaper websites. The figure below shows the sample output using the tool.
2.2.8. Duplchecker

Duplchecker.com provided Unicode supported online tool for plagiarism detection tool that can work with Punjabi text. The software to some extent detects the plagiarism in Punjabi Text, however it does not provide proper comparison and misses out some words while highlighting the source document as show in figure.
Figure 21. Duplichecker highlights the matching source

Figure 22. No plagiarism detected for re-written test