IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 09, Issue 1 (January. 2019), ||V (II) || PP 29-32

Plagiarism Detection: Gaps and Proposed Architecture

Navneet kumar[1], Dr. Meenskshi Sharma[2],
[1]*Ph.D. Research Scholar Singhania university, Pacheri bari Jhunjhunu (Raj.)*
[2]*Associate Professor, Chandigarh University Gharuan (Punjab)*
Corresponding Author: Navneet kumar

Abstract: This paper exposes the flaws in existing plagiarism detection tools. Also, a new architecture has been proposed which tries to cover up the gaps between existing systems and proposed system.

Date of Submission: 28-12-2018 Date of acceptance: 14-01-2019

I. INTRODUCTION

There are various plagiarism detection tools available for processing text in different languages. But none reveals their methodology and architecture. Turnitin[1] and Urkund[2] are the software which checks internet resources as well as alocal repository for plagiarism. But Turnitin doesn't work for Hindi text. It shows a similarity score of 0% for a piece of Hindi text taken from news online. Plagiarism Detect[3], Plagiarism Checker[4], Duplichecker[5], Quetext[6], Antitwin[7] work for Hindi text, but don't consider insertion or deletion of irrelevant words. Also, if a word is substituted by its synonym they don't consider it as plagiarism. Mostly tools work on the syntactic information on the text. Hindi text with modifications was given to the above mentioned tools to obtain the similarity score. And the results categorized the tools into following cases:

Case 1: Tools Unable to Recognize Hindi Text
Various available tools like PlagScan[8]and Plagiarism Checker X[9]failed to recognize Hindi text. Neither they display the Hindi text appropriately, nor are the results appropriate.

Case 2: Incorrect Similarity Score Calculated for Hindi Text for Exact Match
Turnitin is the most popular and most widely used plagiarism detection tool. It gives satisfactory results for English text, as Turnitin checks the suspected document against a very big local repository of English documents as well as against documents available on the Internet. But the system has been found unsuccessful when it is provided the Hindi text as input.

Case 3: Tools Showing Incorrect Results for Text Modified Slightly
Plagiarism Checker calculates the similarity score to be 100% when the documents are exactly same. But when the text has been modified slightly by removing some immaterial words and by replacing some words with their synonyms, the similarity score dropped, which shows Plagiarism Checker does string matching only. The small fragments which have been changed are testified as unique text. Only the exact copy is detected as plagiarized text.

Considering the above cases, it can be concluded that a plagiarism detection software which can identifythe Hindi text and calculates the accurate similarity if the two documents match exactly or even if they vary slightly needs to be developed. So far the focus has been on lexical and structural similarity in natural language, but these become less effective when paraphrasing or data fabrication is done. So the objective is to build a software tool that detects plagiarism, even if there is insertion, removal or substitution of words.

II. PROPOSED ARCHITECTURE

The System has following modules:
i. Tokenization and Stopword Removal
ii. Stemming
iii. Conversion into N-gram
iv. Synonym Replacement
v. Calculation of if-idf and Similarity Score

Mainly, the system has been alienated into two components: Language Independent and Language dependent. Stop word removal, Stemming and Synonym replacement is language dependent modules, whereas Conversion into N-grams and Calculation of if-idf and Similarity score are language independent modules.
The immeasurable complexity of the Natural Language jointly with space and processing time constraint of the computers has raised the necessity for the reduction of the amount of information to be processed by the computational algorithms. This requirement showed the way to the design of preprocessing techniques that reduced the linguistic input while losing the least amount of semantic information. These preprocessing techniques not only paced up the algorithms, but they even improved their performance in many NLP tasks [10]. The preprocessing techniques which are taken up in this design are explained in 2.1 and 2.2.

2.1 TOKENIZATION AND STOP WORD REMOVAL (SWR)
Tokenization is the task of chopping the text into pieces. This is the very first stage of text processing. After tokenization, all the punctuation marks like , and ! are separated from the text. The output of this stage is a representation of the document as a stream of terms.
Stop word removal is a basic pre-handling approach that expels non-content words. Its essential utilize is to keep the handling of text being over-affected by frequent words. Such words incorporate articles, relational words, and other capacity words. These are rather noise, which may diminish the exactness, and in this way it is liked to evacuate them.
List of stop words was taken from [] which was calculated on the basis of frequency and then manual modifications. The list consists of 205 words.

2.2 STEMMING
Stemming is the procedure for decreasing inflected words to their stem. The fundamental motivation behind stemming is to decrease diverse syntactic structures/word types of a word like its thing, descriptive word, action word, qualifier and so on to its root frame. Amid this procedure, the setting of the word is utilized to decide the word sense. This would then be able to be utilized to choose a suitable base frame. Stemming is broadly utilized in Information Retrieval framework and decreases the reduces the size of index files. The objective of stemming is to decrease inflectional structures and some of the time derivationally related types of a word to a typical base frame. Numerous stemmers have been created for various dialects utilizing distinctive methodologies.

Dictionary Based Technique: In the Dictionary based stemmers, every word is matched with a word in a proper digitalized word list which corresponds to its stem. Despite of the method being effective it is not adequate as it is very difficult to deal with unlimited words and their formation. Dictionary-based stemmers require dictionary maintenance, to keep up with an ever-changing language, and this is actually quite a problem.

Rule Based technique: Using some specific rules for the English language, this algorithm removes iteratively suffixes from a given word, reducing it to its stem. The first stemmer developed by Lovin's stemmer [11] follows rule-based technique. Later Porter [12] used this technique to modify the stemmer. He applied the rules iteratively to get the root word. Paice& Husk [13] used the same technique with different rules for their stemmer. A Hindi Rule-based Stemmer has been developed by Gupta [14] for nouns. A list of 16 suffixes has been recognized for nouns. The accuracy of stemmer is 83.65%.

Prefix and Suffix Stripping: It reduces the words to their stem by stripping off the prefixes and suffixes by using a list of provided suffixes. But this stemmer faces problems of over-stemming and under-stemming. Hindi stemmer developed by Anantha KrishnanRamanathan and Durgesh D Rao [15] used suffix stripping algorithm based on a set of rules. An accuracy of 88% is achieved.

A stemmer for the Punjabi language has been developed by Kumar and Rana [16]. This stemmer uses the hybrid approach. This stemmer has combined suffix stripping technique with brute force technique. The root word is searched in a table which consists of a root with their inflected words. If the match is found, then the root word is generated otherwise suffix stripping is done. The stemmer minimizes the problem of over-stemming and under-stemming because of this hybrid approach. 81.27% accuracy is achieved.
Punjabi Stemmer developed by Puri and Goyal [17] uses extended rule set, which stems all categories of Punjabi words. A suffix list is created after careful manual inspection of the Punjabi word inflections. A named entity database, along with Punjabi WordNet is used here to skip unnecessary stripping of named entities. Also, an exceptions list is built to avoid stripping of words which were identified as false positives after stripping

Hindi Stemmer developed by Mishra and Prakash[18] uses combined lookup approach with prefix and suffix stripping. Prefix and suffix removal work on a predefined set of rules. They show theaccuracy of 91.59%. Similar kind of approach has been used by Dogra et al.[9] for developing stemmer for Devanagari script. They showtheaccuracy of 94.26%. The drawback of the stem is extra storage space and manual work due to alookup table.

2.3 CONVERSION INTO N-GRAMS

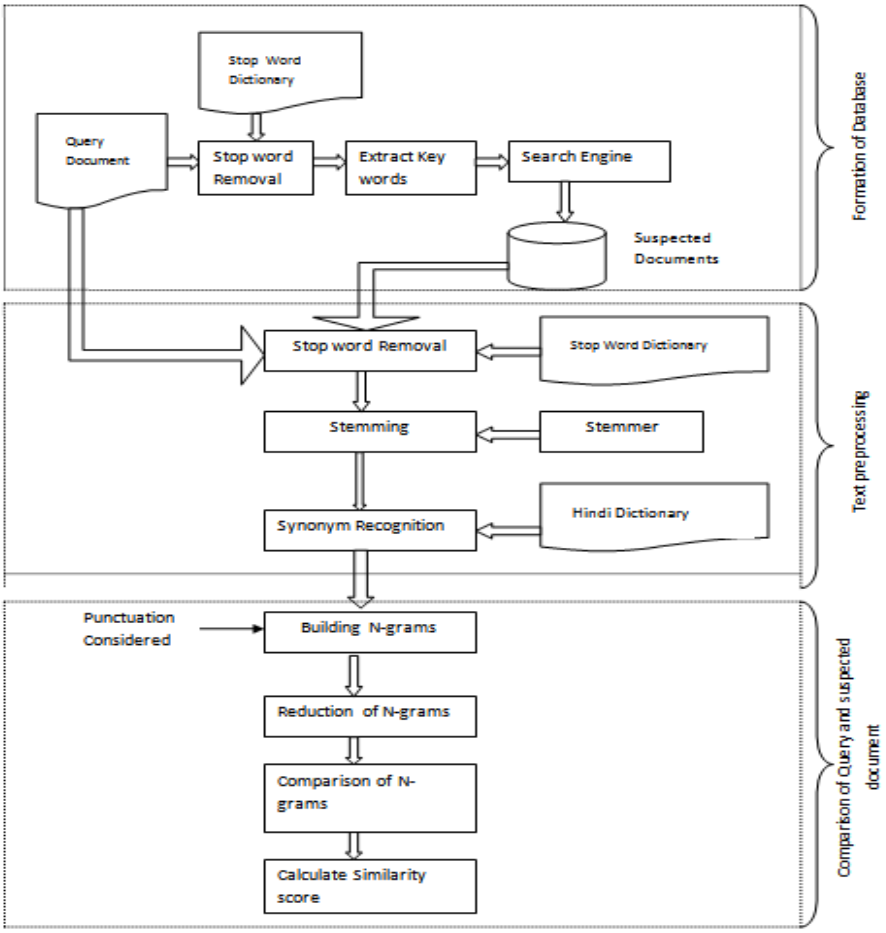
In the fields of computational linguistics and probability , a N-gram is a contiguous grouping of n things from a given succession of content or discourse. The things can be phonemes, syllables, letters, words or base sets as indicated by the application. The N-grams ordinarily are gathered from a content or discourse corpus. A N-gram of size 1 is alluded to as a "unigram"; measure 2 is a bigram, estimate 3 is a "trigram". Bigger sizes are now and then alluded to by the estimation of N, e.g., "four-gram", "five-gram, etc [119]. At the point when the estimation of N is taken as 1, the content is treated as abag of words.

2.4 SYNONYMY RECOGNITION (SYR)

The inspiration for utilizing synonymy acknowledgment originates from thinking about human conduct, where individuals may look to shroud copyright infringement by supplanting words with proper equivalent words. It outwardly changes sentences at first sight, however the structure is left unmodified. In the event that an adequate number of words are supplanted by equivalent words, the majority of the regular duplicate identification strategies come up short. Notwithstanding the highlights the techniques utilize, the best arrangement is to change words having the equivalent or firmly related importance onto an exceptional identifier. Hindi WordNet [21] from IIT, Bombay has been utilized in this work to discover the equivalent words. It comprises of equivalent words of roughly 38000 words and aggregate words are approx1,500,00.

2.5 Calculation of Similarity Score

A similarity measure is a function which computes the degree of similarity between a pair of text objects. There are a large number of similarity measures proposed in the literature. All similarity measures should map to the range [0,1] where 0 shows the minimum similarity and 1 shows the maximum similarity.



III. CONCLUSION

Experiments will be performed on the system developed based on the proposed architecture. We expect a significant difference in the accuracy of previous systems and our system.

REFERENCES

[1]. Turnitin: Leading Plagiarism Detection Tool.Retrieved from http://www.Turnitin.com.
[2]. Urukund.Retrieved from www.urkund.com. Date Accessed: 20th August, 2014.
[3]. Plagiarism Detect. Retrieved fromwww.plagiarismdetect.com. Date Accessed:13thAugust,2011.
[4]. Plagiarism Checker. Retrieved from http://smallseotools.com/plagiarism-checker/. Date Accessed: 17thNovember,2015.
[5]. DupliChecker.Retrieved from http://www.duplichecker.com/. Date Accessed: 18thNovember,2015.
[6]. Quetext.Retrieved from http://www.quetext.com/. Date Accessed: 18th November, 2015.
[7]. Anti Twin. Retrieved fromhttp://www.anti-twin.com/. Date Accessed: 21st November, 2015.
[8]. PlagScan. Retrieved from http://www.plagscan.com/. Date Accessed: 21stNovember,2015.
[9]. Plagiarism Checker X.
[10]. Retrieved fromhttp://plagiarism-checker-x.en.softonic.com/download#downloading. Date Accessed: 20thNovember,2015.
[11]. Pandey, A K, Siddiqui, T J. (2009). Evaluating Effect of Stemming and Stop-word Removal on Hindi Text Retrieval. In *first International Conference on Intelligent Human Computer Interaction.*20th-23rd January, Allahabad, India,316-326, Springer.
[12]. Lovins, J. B. (1968). Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11(1-2), 22-31.
[13]. Porter, M.F. (1980). An algorithm for suffix stripping, *Program*, 14(3), 130-137.
[14]. Paice, C.D. (1990). Another stemmer. *ACM SIGIR Forum*, 24(3), 56-61.
[15]. Gupta, V. (2014). Hindi Rule based Stemmer for Nouns. *International Journalof Advanced research in Computer Science and Software Engineering*, 4(1), 62-65.
[16]. Ramanathan, A.,Rao, D. (2003). A lightweight stemmer for Hindi. *In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL), on Computational Linguistics for South Asian Languages (Budapest, Apr.) workshop.*15th-17th April, Hungary, 42-48.
[17]. Kumar, D., Rana, P. (2010). Design and Development of a Stemmer for Punjabi, *International Journal of Computer Applications*, 11(12), 18-23.
[18]. Puri, R., Bedi, R.P.S., Goyal, V.(2015). Punjabi Stemmer using Punjabi WordNet database. *Indian Journal of Science and Technology*, 8(27), 1-5.
[19]. Mishra, U. ,Prakash, C. (2012). MAULIK: An Effective Stemmer for Hindi Language. *International Journal on Computer Science and Engineering (IJCSE)*, 4(5), 711-711.
[20]. Dogra, M., Tyagi, A. , Mishra, U. (2013). An effective stemmer in Devanagari script. *Proc. of the Intl. Conf. on Recent Trends In Computing and Communication Engineering-- RTCCE 2013.*,22-25, doi10.3850/978-981-07-6184-4_05
[21]. Jurafsky,D. , Martin, J.H.,(2014).(n.d.). Ch-4 N-grams. *Speech and Language Processing*. Retrieved from https://lagunita.stanford.edu/c4x/Engineering/CS-224N/asset/slp4.pdf.
[22]. Hindi WordNet. Retrieved from http://www.cfilt.iitb.ac.in/wordnet/webhwn/index.php. Date Accessed: 07/08/2013.

Navneet kumar. "Plagiarism Detection: Gaps and Proposed Architecture." IOSR Journal of Engineering (IOSRJEN), vol. 09, no. 01, 2019, pp. 29-32.