CHAPTER 4: EXPERIMENTS AND RESULTS

The “Uprala – Plagiarism detection in Punjabi Text” software is developed in open source languages. The system backend is coded using PHP as the primary language (version 5, later on upgraded to 5.6.10) whereas, Javascript and Ajax are used for client side scripting and validations.

The entire frontend interface is designed using open source Twitter Bootstrap 2.0 framework. The Twitter Bootstrap is an industry standard frontend development framework built using HTML 5, CSS 3.0, Javascript and Ajax. This framework also provides various in-built navigation menus, table styles, icons and a number of scripts for tasks such as client side validations, dialog boxes presentations, document uploading etc in the web pages.

The local repository is built using MySQL version 4, later on upgraded to version 5.5.42. The innoDB database engine is used as it provides faster indexing and retrieval using relational database system queries.

The experiments were carried on 2.5 GHz Intel Core i5 processor with 10GB of RAM and 500 GB of Harddisk Space, running Mac OS X version 10 in 64 bit mode.

A local repository of 10,000 articles was created from the articles obtained from local Punjabi Newspaper – Ajit. Various pre-processing tools such as stop word removal, stemming, synonym replacement and keyword identification were applied to the documents before storing them in the local database. This one time task saved good processing time while detecting plagiarism in the query documents.

4.1. EVALUATION METRICS

The relevance of records is most important in the Information Retrieval process. The computer systems can only retrieve the records, but cannot ascertain whether the records retrieved are correct or false. So there has to be some mechanism that can detect the
relevance of the records retrieved in the information retrieval process. Plagiarism detection, being an application of Information retrieval process also needs to detect whether the documents detected as source of plagiarism are actually the source of plagiarism. Secondly, we also need to know how many plagiarized documents go undetected. The Precision-Recall metrics and F-measure [71] collectively serve our purpose and provide a reliable measure to check the accuracy of the designed system. These measures are discussed as below –

4.1.1. Precision

Precision is calculated by dividing the number of relevant records retrieved with the sum of irrelevant records and relevant records. The Precision is usually expressed in percentage. If A represents the number of correctly retrieved records (Relevant), and C represents the number of in-correctly retrieved records (Irrelevant) then PRECISION can be defined as –

\[
\text{PRECISION} = \frac{A}{A + C} \times 100\%
\]

4.1.2. Recall

Recall is calculated by dividing the number of relevant records retrieved with the total number of relevant records present in the database. If A represents the number of correctly retrieved records (Relevant), and B represents the number of relevant records not retrieved by the system, then RECALL can be defined as

\[
\text{RECALL} = \frac{A}{A + B} \times 100\%
\]

4.1.3. F-Measure

The accuracy of Precision and Recall scores is measured with the help of F-Measure. It considers collectively the precision P and the recall R of the test to compute the F-Score. The F-measure is calculated as –

\[
F = \frac{2PR}{P + R}
\]
\[ F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \]

In case, when the PRECISION and RECALL are equally important, then \( \beta = 1 \)

Also the \( F_1 \) value will be calculated as

\[ F_1 = \frac{2PR}{P + R} \]

In information retrieval tasks, the precision score of 1.0 means that every record retrieved by a search was relevant (without giving information on whether all the relevant records were retrieved) whereas a recall score of 1.0 means that all relevant records were retrieved by the search (without giving information on how many irrelevant records were also retrieved in the process).

4.2. EVALUATING PUNJABI STEMMER RESULTS

The stemming results were evaluated using Precision, Recall and F-Measure. A special stemming evaluation module was designed for quick analysis and recording of the stemming process. The stemming module accepts the Punjabi text in Unicode format.

The text is first cleaned up for any punctuation marks, numbers and special characters. The Unicode text is then passed to the stemming module. The module is specially designed with facility for quickly recording the overstemming, understemming and correct stemming results.

The text is evaluated manually, and for each correct stemming, the button with label “C+” is pressed that increments the counter of correctly stemmed words. Similarly, for understemming and overstemming, “U+” and “O+” buttons are used. The complete system is shown in the figure below –
Figure 25. Sample text given for testing stemming results

Figure 26. Sample output of stemming with color coded text
From the report above, it is evident that –

- A total of 492 words were analysed.
- Out of total 492 words, 52 named entities were detected by the system, which were not stemmed.
- Out of total 492 words, 14 words were found in the exceptions table, which were getting over stemmed and the system prevented it from over stemming.
- Total 160 words were found, for which no applicable stemming rule was found.
- Total 92 words were correctly stemmed, which were confirmed from the dictionary fallback.
- Total 39 words were such that which were candidate for stemming due to one or the other rule, but the corresponding stemmed word was not found in dictionary. So the system skipped it.

Figure 27. Legends used for stemming tool, with results
- Total 149 words were ignored due to the reason that their length was below the threshold value.

![Pie chart for quick analysis of Stemmer Results](image)

Figure 28. Pie chart for quick analysis of Stemmer Results

As it is evident from the chart, the stemming results indicate that 36.4% of the words submitted in the text were not stemming as there was no corresponding stemming rule defined for their stemming. A large sum of words (33.9%) were found below the threshold value and hence were too small to stem. 20.9% words were identified as stemming candidates and were correctly stemmed. Out of the total text, 8.9% words were those, for which stemming rules were present, but the stemmed words were not present in the dictionary. Hence the algorithm identified them as incorrectly stemmed and the words were replaced back to their original values.
The analysis shows that only top 10 rules were used for stemming 71.74% of total stemmed words.

Figure 29. Frequency analysis of the Most frequently hit rules for stemming
Stemming With NER

Article: 15329

The text after stemming:

SHOW

Analysis:

Total number of words = 482
Named Entities found = 52
Exceptions found = 14
Not stemmed = 160
Incorrectly stemmed = 30
Too small to stem = 148

Figure 30. Full screen view of stemming tool
The table below shows the experimental Precision, Recall and F-measure for the evaluation of the result –

**Table 10. Stemming Evaluation using Precision, Recall and F-measure**

<table>
<thead>
<tr>
<th>Sr #</th>
<th>Corpus Words</th>
<th>Stemmed Words</th>
<th>Correct Stemmed</th>
<th>Under Stemmed</th>
<th>Over stemmed</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>179</td>
<td>44</td>
<td>36</td>
<td>2</td>
<td>6</td>
<td>85.71</td>
<td>94.74</td>
<td>90.00</td>
</tr>
<tr>
<td>2</td>
<td>290</td>
<td>42</td>
<td>40</td>
<td>2</td>
<td>1</td>
<td>97.56</td>
<td>95.24</td>
<td>96.39</td>
</tr>
<tr>
<td>3</td>
<td>246</td>
<td>44</td>
<td>34</td>
<td>4</td>
<td>6</td>
<td>85.00</td>
<td>89.47</td>
<td>87.18</td>
</tr>
<tr>
<td>4</td>
<td>268</td>
<td>49</td>
<td>44</td>
<td>1</td>
<td>4</td>
<td>91.67</td>
<td>97.78</td>
<td>94.62</td>
</tr>
<tr>
<td>5</td>
<td>209</td>
<td>24</td>
<td>16</td>
<td>2</td>
<td>2</td>
<td>88.89</td>
<td>88.89</td>
<td>88.89</td>
</tr>
<tr>
<td>6</td>
<td>140</td>
<td>27</td>
<td>22</td>
<td>2</td>
<td>3</td>
<td>88.00</td>
<td>91.67</td>
<td>89.80</td>
</tr>
<tr>
<td>7</td>
<td>284</td>
<td>51</td>
<td>43</td>
<td>5</td>
<td>3</td>
<td>93.48</td>
<td>89.58</td>
<td>91.49</td>
</tr>
<tr>
<td>8</td>
<td>273</td>
<td>84</td>
<td>71</td>
<td>3</td>
<td>10</td>
<td>87.65</td>
<td>95.95</td>
<td>91.61</td>
</tr>
<tr>
<td>9</td>
<td>544</td>
<td>79</td>
<td>67</td>
<td>6</td>
<td>6</td>
<td>91.78</td>
<td>91.78</td>
<td>91.78</td>
</tr>
<tr>
<td>10</td>
<td>268</td>
<td>67</td>
<td>61</td>
<td>2</td>
<td>5</td>
<td>92.42</td>
<td>96.83</td>
<td>94.57</td>
</tr>
<tr>
<td>11</td>
<td>151</td>
<td>20</td>
<td>17</td>
<td>1</td>
<td>2</td>
<td>89.47</td>
<td>94.44</td>
<td>91.89</td>
</tr>
<tr>
<td>12</td>
<td>251</td>
<td>36</td>
<td>33</td>
<td>1</td>
<td>2</td>
<td>94.29</td>
<td>97.06</td>
<td>95.65</td>
</tr>
<tr>
<td>13</td>
<td>93</td>
<td>17</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>93.75</td>
<td>93.75</td>
<td>93.75</td>
</tr>
<tr>
<td>14</td>
<td>111</td>
<td>28</td>
<td>21</td>
<td>2</td>
<td>5</td>
<td>80.77</td>
<td>91.30</td>
<td>85.71</td>
</tr>
<tr>
<td>15</td>
<td>170</td>
<td>34</td>
<td>39</td>
<td>0</td>
<td>4</td>
<td>90.70</td>
<td>100.00</td>
<td>95.12</td>
</tr>
<tr>
<td>16</td>
<td>216</td>
<td>40</td>
<td>35</td>
<td>1</td>
<td>4</td>
<td>89.74</td>
<td>97.22</td>
<td>93.33</td>
</tr>
<tr>
<td>17</td>
<td>244</td>
<td>39</td>
<td>36</td>
<td>0</td>
<td>3</td>
<td>92.31</td>
<td>100.00</td>
<td>96.00</td>
</tr>
<tr>
<td>18</td>
<td>220</td>
<td>37</td>
<td>36</td>
<td>1</td>
<td>0</td>
<td>100.00</td>
<td>97.30</td>
<td>98.63</td>
</tr>
<tr>
<td>19</td>
<td>312</td>
<td>70</td>
<td>63</td>
<td>2</td>
<td>5</td>
<td>92.65</td>
<td>96.92</td>
<td>94.74</td>
</tr>
<tr>
<td>20</td>
<td>432</td>
<td>89</td>
<td>80</td>
<td>1</td>
<td>8</td>
<td>90.91</td>
<td>98.77</td>
<td>94.67</td>
</tr>
</tbody>
</table>
4.3. STOP WORD REMOVAL

The words account for the huge size of the corpus. These words should be removed in the preprocessing phase of the text classification process. This will not only decrease the vector space of the corpus, but will also speed up the calculations and increase the accuracy of the IR task. A frequency analysis of the words from the repository of 10,000 articles obtained from Ajit Newspaper [55] with approx. 400 words per article was prepared after skipping the punctuation marks, white spaces and other non-Punjabi characters. The figure below shows the stop words removal tool in action –

Figure 31. Stop words Removal tool showing %reduction in corpus size
On experimentation with different size of articles, it was found that an average reduction of 49.5% was obtained as shown in the table below –

**Table 11. Effect of Stop word Removal in size of corpus**

<table>
<thead>
<tr>
<th>No. of Tokens in Text</th>
<th>Reduction in %age</th>
</tr>
</thead>
<tbody>
<tr>
<td>307</td>
<td>54</td>
</tr>
<tr>
<td>418</td>
<td>39</td>
</tr>
<tr>
<td>465</td>
<td>48</td>
</tr>
<tr>
<td>504</td>
<td>54</td>
</tr>
<tr>
<td>762</td>
<td>40</td>
</tr>
<tr>
<td>960</td>
<td>57</td>
</tr>
<tr>
<td>1219</td>
<td>55</td>
</tr>
<tr>
<td>1319</td>
<td>51</td>
</tr>
<tr>
<td>1528</td>
<td>50.1</td>
</tr>
<tr>
<td>2077</td>
<td>47</td>
</tr>
<tr>
<td>Average Compression</td>
<td>49.5%</td>
</tr>
</tbody>
</table>

### 4.4. KEYWORDS IDENTIFICATION

The keywords retrieved from a document help in building a list of suspected documents, where there are high chances of finding the matching contents. The suspected documents filtration helps reducing the document vector space and speed up the plagiarism detection process. The TF-ISF approach [63] is used in our system to find the important keywords related to the document. The figures below the keyword identification task in action. The text is pasted in the keyword detection module. The module calculates the TF-ISF values as per the algorithm discussed in chapter 3. The calculated values of TF-ISF per keyword are shown below the keyword so as to quickly have a glance on the working of the module. The module also has been designed to optionally use the stemmer developed in the previous step before calculating the TF-ISF values of the keywords.
Figure 32. Keyword Detection Module - Without word stemming

Figure 33. Keyword Detection Module - With word stemming
Figure 34. Keywords Detection - 20 keywords with Min. occurrence 2 times, without stemming

Figure 35. Keywords Detection - 20 keywords with Min. occurrence 2 times, with stemming
With experimental results, it was evident that the stemming improved the keyword detection process by reducing the inflated words to their root words. For example, the keywords ਅ ਤਵ ਦ ਆ has a greater TF-ISF value as compared to the keyword ਅ ਤਵ ਦ. However, the word ਅ ਤਵ ਦ ਆ is basically an inflected form of the word ਅ ਤਵ ਦ. The developed module takes care of these by stemming the keyword ਅ ਤਵ ਦ ਆ to ਅ ਤਵ ਦ while calculating the TF-ISF score.
4.5. N-GRAM SIZE AND COSINE SIMILARITY

In an experimental setup, a sample corpus was manually crafted from 50 random articles by selectively copying the text from five articles only. The Cosine Similarity was calculated by varying the size of n-grams. Also the time taken for the similarity calculation was recorded.

Figure 37. Similarity Calculation of Sample Punjabi Text document

The diagram above shows the number of documents used from the repository for calculation of the similarity. In this particular experiment,

- Repository size of 50 documents was used.
- Trigrams were created out of the available text.
- The system took 6.12 seconds for calculation of the similarity score.
- The match of 42% was found from four documents.
- Each match was displayed using a separate color coding.
Figure 38. Similarity calculation vs time taken with variation in nGram size

The diagram above shows the number of documents used from the repository for calculation of the similarity. In this particular experiment,

- Repository size of 50 documents was used.
- A total of 4960 tokens were created out of the available text.
- The system took 6.12 seconds for calculation of the similarity score.
- The match was found from five documents.
In next experiment, the nGram size of 2 was used. The diagram shows the number of documents used from the repository for calculation of the similarity. In this particular experiment,

- Repository size of 50 documents was used.
- A total of 11761 tokens were created out of the available text.
- The system took 14 seconds for calculation of the similarity score.
- The match of 61% was found from seven documents from the repository.
In next experiment, the nGram size of 3 was used. The diagram shows the number of documents used from the repository for calculation of the similarity. In this particular experiment,

- Repository size of 50 documents was used.
- A total of 12472 tokens were created out of the available text.
- The system took 16.35 seconds for calculation of the similarity score.
- The match of 35.8% was found from two documents from the repository.
It was found that the more precise results were obtained with n-grams of size 4.

![Figure 41. Effect of n-gram size on Cosine Similarity](image)

4.6. STEMMING AND COSINE SIMILARITY

The effect of stemming on cosine similarity was evaluated and it was found that the cosine similarity decreased when stemming was not used in similarity calculations (Figure 42). N-gram size 4 was chosen for the calculations in both cases.

![Figure 42. Effect of Stemming on Cosine Similarity](image)
4.7. SYNONYM REPLACEMENT AND COSINE SIMILARITY

Effect of synonym replacement on cosine similarity was calculated on the same dataset. It was found that cosine similarity decreased when synonym replacement was not used in similarity calculations. N-gram size 4 with stemmer was chosen in both cases (Figure 43).

![Figure 43. Effect of Synonym Replacement on Cosine Similarity](image)

4.8. COMPLETE WORKFLOW

The entire system has been designed to work in online environment. The users need to register first to submit their documents for plagiarism detection. The welcome screen on home page asks the new users to sign up with the software. The returning users however need to provide their unique username and password to login to the system. The screenshot below shows the home page of the software.

The twitter bootstrap [72] is used for the UI design. This bootstrap template provides a framework of several client side validations including verifying the email format in signup and login processes. We can also provide the minimum password lengths to be used in the password fields in bootstrap templates.
The software has been designed to support two types of users –

- **Basic Users** – Can submit their plagiarism detection jobs and see the results
- **Advance Users (Admins)** – Apart from job submissions, can run individual modules for research and development purpose. The advance users are provided with the following modular facilities
  - Build up Dictionary
  - Text Cleaner
  - Unicode conversion
  - Stopword removal
  - nGrams Generator
  - Keywords Detection
  - Stemming
    - With Named Entity detection
    - Without named entity detection
    - Without dictionary verification
Repository Management

- Upload using Carwler
- Upload using PDF/Doc/TXT files
- Manage Repository

The figures below show the various menu options available to the Advance users –

Figure 45. Tools menu showing various Modular tools

Figure 46. Repository Management

The system is provided with a special crawler that can use Google API as well as screen scraping techniques to fetch the contents out of the web page. The URLs obtained from the search engine was crawled and screen scraped to temporarily build up the repository for the comparison purpose. The module also allows to view, modify and delete the contents of the repository.
Figure 47. Stemming options

Three separate options were provided to the users to evaluate the stemming techniques –

- Stemming with Dictionary and Named Entity Recognition
- Stemming with Dictionary, but without Named Entity Recognition
- Stemming without Dictionary and Without Named Entity Recognition

Figure 48. Similarity Detection for entered text, without creating a job

A separate quick cosine similarity calculation module was developed that just calculates the similarity score for quick reference.
The dictionary management module allows for the modification to the dictionary words. The administrators can add new words to the dictionary, remove existing words from the dictionary as well as edit the words present in the dictionary.

Figure 49. Dictionary Management module
<table>
<thead>
<tr>
<th>Job ID</th>
<th>Title</th>
<th>Submitted on</th>
<th>Completed on</th>
<th>Repository Result</th>
<th>Internet Result</th>
<th>Force Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>Single</td>
<td>2016-07-31 08:45:00</td>
<td>2016-07-31 20:31:00</td>
<td>Local</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>Mix 6</td>
<td>2015-07-31 08:33:00</td>
<td>2016-07-31 20:56:00</td>
<td>Local</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Mix 5</td>
<td>2015-07-31 07:39:00</td>
<td>2016-07-31 20:57:00</td>
<td>Local</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>Mix article</td>
<td>2016-07-31 04:20:00</td>
<td>2016-07-31 18:35:00</td>
<td>Local</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>Test article</td>
<td>2016-07-31 04:22:00</td>
<td>2016-07-31 17:48:00</td>
<td>Local</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Test article</td>
<td>2016-07-31 04:31:00</td>
<td>2016-07-31 16:07:00</td>
<td>Local</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Figure 50. Jobs board, displaying list of jobs submitted along with results

Figure 51. Plagiarism detection results for the job submitted by the user
Figure 52. Submitting new job for plagiarism detection

Figure 53. Real time plagiarism detection in action
4.9. COMPARISON WITH OTHER AVAILABLE TOOLS

A comparison of the effectiveness of the software was done by preparing special articles in Punjabi using various internet sources, first by copying the exact paragraphs from the documents and second by making changes to those articles by using paraphrasing, synonym replacements and sentence reordering. The given document created from random articles (Original articles available on Internet) was fed to the plagiarism detection tools available on the date. The following result sets were obtained as per the similarity calculations done by the software.

Table 12. Comparison of Plagiarism Detection tools

<table>
<thead>
<tr>
<th>Sr#</th>
<th>Tool</th>
<th>Similarity (Original Text)</th>
<th>Similarity (Altered Text)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>TurnItIn</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>2.</td>
<td>Urkund</td>
<td>100%</td>
<td>81%</td>
</tr>
<tr>
<td>3.</td>
<td>PlagScan</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>4.</td>
<td>Small SEO Tools Plagiarism Detection</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>5.</td>
<td>PlagiarismSoftware.net</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>6.</td>
<td>QueText</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>7.</td>
<td>Duplichecker</td>
<td>65%</td>
<td>0%</td>
</tr>
<tr>
<td>8.</td>
<td>wCopyFind</td>
<td>81%</td>
<td>0%</td>
</tr>
<tr>
<td>9.</td>
<td>Uprala</td>
<td>100%</td>
<td>96%</td>
</tr>
</tbody>
</table>

It was observed that the Uprala software outperformed all other tools when posed with the Punjabi contents. The graph below elaborates the data in the table given above.
Figure 54. Graphical Presentation of Similarity Scores Detected by Various Plagiarism Detection Software.