PROJECT ABSTRACT

Plagiarism Detection System using NLP

Protecting academic integrity has never been harder.

01 Title

Plagiarism Detection System using NLP

02 Introduction / Background

Protecting academic integrity has never been harder. Assignments can be assembled from copied sources in minutes, while commercial checkers charge per-seat licenses many departments cannot afford. Faculty lack a tool to verify originality at scale. Similarity detection — preprocessing, vector-space models, fingerprinting — is among the instructive NLP topics, making a working detector an ideal final-year project.

03 Problem Statement

Colleges need in-house similarity checking but cannot depend on commercial services requiring uploads to third-party servers. Naive string matching fails against paraphrased or reordered copying — how plagiarism is done. OrigCheck solves both: an engine combining TF-IDF cosine similarity with winnowing fingerprinting, in a Flask app with PDF reporting.

04 Objectives

- Similarity engine comparing one submission against a full corpus, at document and passage level.
- TF-IDF cosine similarity plus winnowing-based fingerprinting.
- Originality percentage, per-source breakdowns, side-by-side matched-passage highlighting.
- Formal originality report exported as PDF for department records.
- Configurable threshold, n-gram sizes, winnowing settings; evaluated on planted plagiarism.

05 Proposed System / Working

NLTK preprocessing first: tokenization, lowercasing, stopword removal and stemming. Two techniques run in parallel. First, TF-IDF: TfidfVectorizer converts each document into a vector; cosine similarity measures overlap with every corpus document. Second, winnowing: a module hashes text into k-grams and selects a fingerprint via a sliding-window minimum — catching reordering and paraphrasing that vector comparison misses. Matched passages are highlighted side by side with similarity bands. The OrigCheck Flask app handles corpus management, checking, and PDF reports.

In demo, two assignments are uploaded and matches light up instantly — verbatim paragraphs

flagged by cosine similarity, a reordered passage caught by the winnowing fingerprint. On the sample corpus with planted plagiarism, the engine flags copied passages at the threshold. Threshold tuning demonstrates the precision-recall trade-off — an examiner favourite.

06 Technologies / Components Used

Python 3 · **NLTK** · **scikit-learn** · **Custom winnowing fingerprinting implementation** · **Flask** · **HTML/CSS/JS frontend** · **WeasyPrint** · **SQLite**

07 Key Features

- Multi-document checking: one submission against an entire corpus.
- TF-IDF cosine similarity at document and passage level.
- Winnowing fingerprinting catching paraphrased, reordered and lightly edited copying.
- Side-by-side matched-passage highlighting with similarity bands.
- Per-source similarity percentages plus an overall originality score.
- Downloadable originality report (PDF) with matched excerpts and source references.
- Configurable similarity threshold and n-gram / shingle sizes.

08 Applications / Use Cases

- OrigCheck suits college departments checking assignments, lab reports and theses — replacing per-seat licenses while keeping student work on local servers.
- The same engine applies to publishers checking manuscripts and platforms screening submissions.

10 Conclusion

From NLP theory to a tool colleges could deploy: building TF-IDF similarity and winnowing fingerprinting from first principles, backed by evaluation data, delivers the depth examiners reward and a working product with clear value.