Dravidian Language Identification Model

Sruthi Santhanam, Yashvardhan Sharma

Department of Computer Science and Information Systems, Birla Institute of Technology and Science Pilani, Pilani Campus

Abstract

In this paper, a word-level language identification system is presented for the Dravidian languages Tamil, Telugu, Malayalam, Kannada, and Tulu using a combination of TF-IDF character n-grams, handcrafted features, and FastText embeddings. These languages have rich morphological structures and diverse scripts, making computational processing challenging. For each language, there is a training set in CSV format with many words and their labels (e.g., tam for Tamil, en for English, Location, Name, symbol, etc.). The test set contains only words, and the goal is to predict the labels. This approach combines multiple types of features, which are TF-IDF on character n-grams, handcrafted features, and FastText embeddings, which are given as input to a linear SVM LinearSVC, which gives the predictions as output. The validation set evaluates accuracy. Results show strong performance across all five languages, demonstrating that integrating statistical and embedding-based features is effective for Dravidian language identification.

Keywords

Language Identification, FastText, TF-IDF, SVM

1. Introduction

Dravidian languages have beautiful and complex word structures that make automatic language processing difficult [5]. Many of these languages lack digital resources, making tools to analyse them challenging but important [7][8]. These languages are a significant part of Indian, specifically South Indian, culture that needs to be preserved. This project aims to create a machine learning system that can predict linguistic labels for words in different Dravidian languages. The main Dravidian Languages are Tamil, Telugu, Malayalam, Kannada, and Tulu. From a computer science point of view, language identification is a classification problem where the goal is to automatically determine the language of a given word or text. It is an important part of many NLP applications such as translation, information retrieval, and text classification.

The same machine is applied to all 5 main languages for evaluation, incorporating previous work in word-level language identification. The first step is pre-processing, where the training and validation sets are made clean. Rows with empty words are removed from both sets so that the model trains only on valid words and their labels. The next step is feature extraction. Three types of features are incorporated: TF-IDF on character n-grams, handcrafted features, and FastText embeddings. Character-level TF-IDF n-gram features detect patterns in suffix, prefix, or common root that frequently occur in Dravidian languages. Handcrafted features such as word length, capitalization, and the presence of digits or special characters provide hints at the characteristics of the word. FastText is an open-source library developed by Facebook AI Research for text classification. It works not only on each word but also on subwords to be more precise. The output from combining these features is fed into a Linear Support Vector Machine (LinearSVC), a fast and accurate classifier for high-dimensional data. The classifier assigns the ideal weight to each of the features by learning during training (with a maximum of 10,000 iterations). The machine uses these weights on the test words and gives their predicted labels as output. This model follows a machine learning approach rather than a rule-based or purely statistical model because it can learn useful language patterns directly from data instead of relying on manually crafted linguistic rules. Machine learning also allows easy extension of the same pipeline to multiple languages and updates it whenever new data becomes available.

Forum for Information Retrieval Evaluation, December 17-20, 2025, India

f20220837@pilani.bits-pilani.ac.in (S. Santhanam); yash@pilani.bits-pilani.ac.in (Y. Sharma)

2. Related Work

Research on word-level language identification for Dravidian languages has grown steadily in recent years, driven by the need to handle code-mixed and morphologically rich text. Benchmark datasets and shared tasks, such as the CoLI-Dravidian 2025 challenge, have provided a unified framework for evaluating models across Tamil, Telugu, Malayalam, Kannada, and Tulu, revealing the difficulties of rare labels, multiple scripts, and complex word structures [13]. Prior efforts often focused on individual languages, including Tulu-English and Kannada-English code-mixed corpora, which enabled the development and testing of machine learning approaches combining n-gram statistics, handcrafted linguistic features, and embedding-based representations [14]. Beyond shared tasks, the creation of annotated corpora for under-resourced Dravidian languages has played a key role in advancing research. Such datasets not only support language identification but also facilitate related tasks like sentiment analysis and named entity recognition, providing a foundation for models to capture both frequent and rare patterns in the data [14]. Collectively, these efforts have established the resources, evaluation methodologies, and feature engineering strategies that inform the design of the current Dravidian language identification model, allowing it to effectively handle multiple languages within a single unified framework.

3. State Of The Art

Language identification at the word level in multilingual and code-mixed contexts is an active area of research in natural language processing. As mentioned in the previous section, for Dravidian languages, which have complex structures and appear frequently in code-mixed data, shared tasks establish benchmark datasets and evaluation frameworks that highlight the challenges of handling multiple scripts and rare labels [1]. Earlier methods often rely on statistical language models and n-gram analysis, which provide reasonable results but struggle with rare tokens [2]. More recent approaches introduce machine learning classifiers such as Support Vector Machines, which perform well on character-level features in classification tasks [12]. Representations like TF-IDF are also widely adopted, as they capture orthographic patterns including prefixes, suffixes, and common roots [11]. Handcrafted features, such as capitalization, word length, and digit counts, are sometimes added to detect named entities and specific token types. Embedding-based methods bring further improvements by incorporating subword information. This allows models to create meaningful vector representations for unseen or rare words, which is particularly useful for under-resourced Dravidian languages. FastText embeddings, which combine character-level subword information with word-level vectors, are a common choice for this task [9].

4. Task Description

There are five subtasks under this task, each corresponding to a different Dravidian language. A training and validation set for each language contains words and their labels, and a test set contains words only. The goal is to predict the labels for the words in the test dataset, which is considered the run submission taken for evaluation. The shared task for this year includes all five Dravidian languages, each representing a separate subtask with its corresponding dataset. This differs from previous editions, which focused on individual languages separately rather than addressing all of them simultaneously. The labels are unique to each language, and some labels are challenging to predict precisely as they are rare in the training set. For example, there is no Location label for the words in the Malayalam training and validation datasets, but it is called Place instead. A maximum of five run submissions is allowed for each language.

5. Methodology

As aforementioned, the same algorithm is used for all five languages. In this description of methodology, the Tulu language is used as a reference to explain the process. There are three main steps in this algorithm, which are pre-processing to clean the data, Feature engineering, including three types of features, and Prediction using LinearSVC. The design of the model is consistent across languages, which ensures that the same pipeline of preprocessing, feature extraction, and prediction can be applied without major modifications.

5.1. Pre-Processing

The first step of the pipeline is preparing the input data. The training set and validation set are carefully cleaned by removing rows that contain empty words. This prevents the model from training on incomplete entries, which could introduce errors or reduce classification accuracy. After cleaning, the training, validation, and test datasets are loaded into the algorithm. The training set is used to learn feature weights, the validation set is used to measure performance, and the test set is used to generate the final predictions. This structured handling of input data ensures that the model learns only from reliable examples and can generalize effectively to unseen words. The choice of preprocessing all the languages with the same steps is intentional. Each dataset follows a similar CSV format, and basic cleaning operations, such as removing empty words and normalizing text, are applied. The deeper language differences are captured in the later stages, like feature extraction. As these representations automatically adapt to structural variations across languages, the universal preprocessing keeps the pipeline simple.

5.2. Feature Extraction

TF-IDF (Term Frequency-Inverse Document Frequency) on Character n-grams: Each word is broken into small overlapping sequences of 2 to 5 characters (called n-grams). TF-IDF calculates how important each n-gram is for a given word relative to the entire set of words [6]. It captures patterns in spelling like prefixes, suffixes, and common roots that help differentiate words. TF (Term Frequency) is how often an n-gram of a word is repeated in the word itself. For example, in the word “tulu”, the 2-gram tu has a term frequency of 1. DF (Document Frequency) is how many times a particular n-gram occurs in the entire dataset (like the training dataset). To calculate IDF:

$$\text{IDF}(t) = \log\left(\frac{N}{1+n_t}\right)$$

where:

$$N = \text{Total number of words (documents)}$$
$$n_t = \text{Number of words containing n-gram } t$$

N-grams that are common have low IDF, so they are down-weighted, and rare n-grams have high IDF and hence are up-weighted. To calculate TF:

$$\text{TF}(t,d) = \text{Number of times n-gram } t \text{ appears in word (document) } d$$

Alternatively, if normalized frequency is more preferable:

$$\text{TF}(t,d) = \frac{\text{Number of times n-gram } t \text{ appears in } d}{\text{Total number of n-grams in } d}$$

where:

$$t = \text{n-gram}$$
$$d = \text{word (document)}$$

The TF-IDF Score is obtained by multiplying TF and IDF:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Hence, if an n-gram occurs frequently in a given word but rarely in other words, it gets a high TF-IDF score, so it is important in classifying the word uniquely.

Custom Handcrafted Features: A few simple differentiators are added, such as word length, capitalization, alphanumeric check, and digit count. Capitalization is used as it helps identify proper nouns (label = name). Word length is relevant as the number of characters is analyzed, since longer words often have different linguistic roles [3].

FastText Embeddings:

FastText is a tool created by Facebook for representing words as vectors that capture their meaning [10]. These vectors are called embeddings. Each word is mapped into a 300-dimensional vector. Words that are similar in meaning or usage have vectors that are close together in this "embedding space" [4]. FastText breaks each word into subwords (like character n-grams). To get the word's full vector, FastText combines the vectors for its subwords and the whole word. FastText then shows what vectors are similar to this word vector. Hence, even if an unknown or rare word appears (like a new Tulu term), FastText can build an embedding based on subwords, giving it a reasonable meaning. All three of these feature types are combined for each word into one large numeric matrix. This combined feature matrix gives a thorough representation for each word, improving the ability of the model to classify words accurately.

5.3. Prediction

The model is trained using a linear Support Vector Machine. SVM is a powerful machine learning algorithm used for classification when each data point (here, each word) needs to be assigned to one of several categories (labels such as tam, en, sym, etc.) [3]. A Linear SVM specifically tries to find a straight-line boundary (which in higher dimensions becomes a flat plane called a hyperplane) that separates the data points of one class from another in the feature space. SVM looks at all the features of the words and assigns different weights to each of the features to determine the best boundary for grouping the same labels. SVM checks how accurate each set of assigned weights for the features is by comparing it with the actual label in the training set. It learns during training with a maximum number of iterations of ten thousand. If the model converges to a value before ten thousand iterations, it stops and proceeds to the next step. After training, SVM gives a set of weights (one weight for each feature) as output. When predicting a new word's label, SVM looks at the features of the word (output from feature extraction, a combination of the three feature types described above), and it assigns the ideal weight to each feature as learned during training. Hence, SVM classifies the words by repeating the same calculation for the test set. The trained SVM model uses its learned boundaries to assign the most likely tag for each word based on its features. The linear SVM used in this algorithm is implemented using the LinearSVC class from the scikit-learn Python library. Linear SVM is selected as the main classifier after testing simpler baselines such as Logistic Regression and Naïve Bayes. SVM gives the best trade-off between speed and accuracy for high-dimensional TF-IDF and FastText vectors, which require heavy computation. While more complex deep models could be explored in future work, this classifier works well for this lightweight system that performs strongly. The validation set is used to measure performance, and the test set predictions are the final output that is submitted in a zip file for evaluation. A predictions zip file is submitted for each language, and all are evaluated separately, with ranklists published accordingly.

6. Model Architecture

Figure 1 illustrates the complete workflow of the Dravidian Language Identification Model. The process begins with the input data, which includes the training, validation, and test sets. In the pre-processing stage, empty rows are removed, and text is normalized. Three types of features are then extracted: (i) TF-IDF on character n-grams, which captures orthographic patterns, (ii) handcrafted features, including word length, capitalization, and presence of digits, and (iii) FastText embeddings, which provide subword-level semantic representations. These features are concatenated into a single feature matrix and fed into a Linear Support Vector Machine (LinearSVC) for model training. Finally, the trained model generates predictions for the test set, and evaluation metrics such as accuracy, precision, recall, and F1-score are computed. The diagram provides a high-level overview of the pipeline, showing the flow from raw input to final predictions.

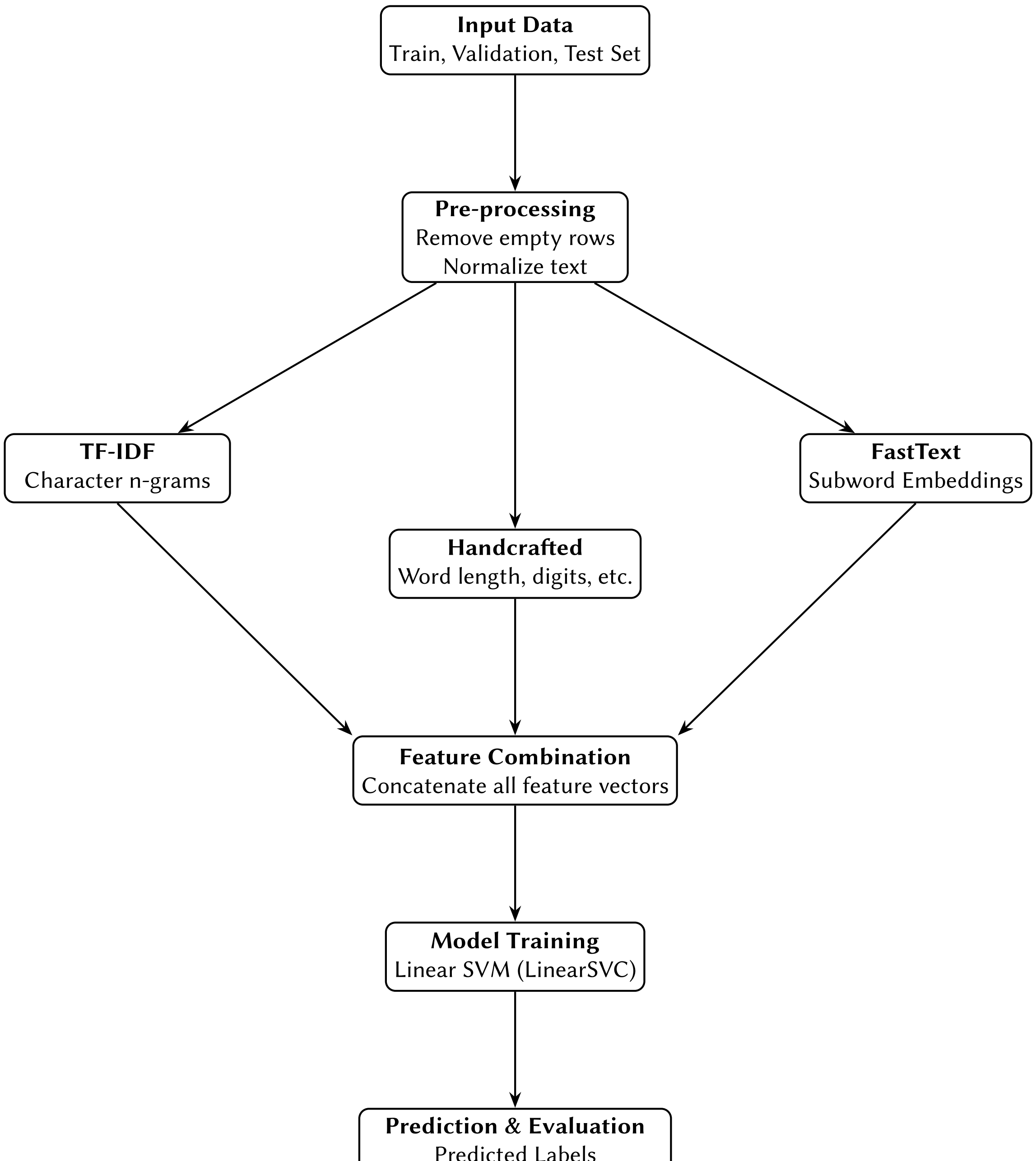

Figure 1: Outline of the Language Identification Model

7. Results

Table 1
Performance Metrics for Dravidian Languages

Rank in WILD	Language	Wtd Precision	Wtd Recall	Wtd F1	Macro Precision	Macro Recall	Macro F1	Accuracy
1	Kannada	0.96	0.96	0.96	0.94	0.90	0.92	0.96
1	Tulu	0.91	0.91	0.91	0.87	0.79	0.82	0.91
2	Tamil	0.92	0.92	0.92	0.79	0.71	0.73	0.92
2	Telugu	0.95	0.96	0.95	0.80	0.80	0.79	0.96
2	Malayalam	0.88	0.88	0.88	0.87	0.79	0.82	0.88

The performance of this algorithm on these five Dravidian languages is shown in the table above. The rank column refers to the rank obtained using this algorithm in the CoLI-Dravidian task held on CodaBench. Wtd in the table refers to Weighted. Overall, the model achieves high accuracy and strong scores on most evaluation metrics across all five languages. Kannada obtains the best results among the languages with an accuracy of 0.96 and a macro F1 score of 0.92, showing that the classification is consistent and reliable. Telugu and Tamil perform well. Malayalam gives the lowest scores among the group. The differences in the performance of the same algorithm across the five languages may be due to various reasons, such as training data size or particular language features. Overall, the combination of features such as FastText embeddings and TF-IDF on character n-grams processed by LinearSVC contributes to good performance, demonstrating that this model classifies Dravidian languages effectively despite their complexity.

8. Discussion

The results show that while the model performs strongly overall, there are still clear areas for improvement. One important observation is that the same feature set does not behave equally across all languages, which suggests that some language-specific tailoring may be necessary. For example, Malayalam performs worse compared to the other languages. The first suspect for this was that the training set for the Malayalam language may have been significantly smaller than those of the other languages. However, this is not the case. The Malayalam training set has 25996 entries, whereas the Tulu training set has 29525 entries. Hence, this is likely an incorrect assumption. The poorer performance of the Malayalam language identification model may be due to different labels being inconsistent in frequency. For example, the label “PLACE” occurs only 123 times in the Malayalam training set, whereas the label “MALAYALAM” occurs 11794 times and “ENGLISH” occurs 5768 times, showing that the label “PLACE” is rare and underepresented. This indicates that adding more balanced datasets or applying data augmentation could improve performance. Another promising direction is the use of deep learning models, such as recurrent or transformer-based architectures, which may capture richer contextual information than handcrafted features alone. At the same time, a hybrid approach that combines TF-IDF and FastText with neural models could strike a balance between interpretability and performance. Finally, improving the handling of rare labels remains an open challenge, and exploring techniques like few-shot learning or external lexical resources may help address this gap.

9. Conclusion

In this task, the development of a machine learning approach for classifying Dravidian words was explored. The model’s performance was evaluated repeatedly during development using the validation dataset. By combining the three different types of features of TF-IDF on character n-grams, handcrafted features, and FastText word embeddings, the model captured the linguistic properties, including both the surface-level and deep information of each word by representing it in a numerical matrix form. LinearSVC was used as the classifier due to its speed and compatibility for differentiating high-dimensional data. Overall, the results were satisfactory. However, the model could be improved by adding more or different layers or by having bigger training datasets with more instances of words with rare labels. Working on this project gave me a better understanding of Language Identification (LI) and the different tools one could use to build a model to tackle the task of LI and NLP in general.

Declaration on Generative AI

In the preparation of this paper, a generative AI tool was used in a limited manner only for paraphrasing. The author(s) utilized QuillBot for rephrasing certain sentences to improve clarity and readability. All conceptual content, implementation details, analysis, and final editing were carried out by the author(s), who take full responsibility for the accuracy and integrity of the work.

References

[1] Hegde, A., Balouchzahi, F., Coelho, S., HL, S., Nayel, H. A., & Butt, S. (2023, December). CoLI@ FIRE2023: Findings of Word-level Language Identification in Code-mixed Tulu Text. In *Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation* (pp. 25–26).

[2] Hegde, A., Balouchzahi, F., Coelho, S., Shashirekha, H. L., Nayel, H. A., & Butt, S. (2023). Overview of CoLI-Tunglish: Word-level Language Identification in Code-mixed Tulu Text at FIRE 2023. In *FIRE (Working Notes)* (pp. 179–190).

[3] Balouchzahi, F., Butt, S., Hegde, A., Ashraf, N., Shashirekha, H. L., Sidorov, G., & Gelbukh, A. (2022, December). Overview of CoLI-Kanglish: Word-Level Language Identification in Code-Mixed Kannada-English Texts at ICON 2022. In *Proceedings of the 19th International Conference on Natural Language Processing (ICON): Shared Task on Word Level Language Identification in Code-mixed Kannada-English Texts* (pp. 38–45).

[4] Lakshmaiah, S. H., Balouchzahi, F., Anusha, M. D., & Sidorov, G. (2022). CoLI-Machine Learning Approaches for Code-mixed Language Identification at the Word Level in Kannada-English Texts. *Acta Polytechnica Hungarica*, 19(10), 123–141.

[5] Sai, S., & Sharma, Y. (2021, April). Towards offensive language identification for Dravidian languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages* (pp. 18–27).

[6] Mandalam, A. V., & Sharma, Y. (2021, April). Sentiment analysis of Dravidian code-mixed data. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages* (pp. 46–54).

[7] Ojo, O. E. (2022). Language Identification at the Word Level in Code-Mixed Texts. In *Proceedings of the 19th International Conference on Natural Language Processing (ICON).*

[8] Hegde, A., Balouchzahi, F., Coelho, S., HL, S., Nayel, H. A., & Butt, S. (2023). Word-level Language Identification in Code-mixed Tulu Text: Shared Task Overview. In *FIRE (Working Notes).*

[9] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (pp. 427–431). Association for Computational Linguistics.

[10] FastText. (2017, October 2). Language identification – fastText. Retrieved from https://fasttext.cc/blog/2017/10/02/blog-post.html

[11] Capital One. (2021, October 6). Understanding TF-IDF for Machine Learning. Capital One Tech Blog. Retrieved from https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

[12] Campbell, W. M., Campbell, J. P., Reynolds, D. A., Singer, E., & Torres-Carrasquillo, P. A. (2006). Support vector machines for speaker and language recognition. *Computer Speech & Language*, 20(2–3), 210–229.

[13] Hegde, A., Balouchzahi, F., Butt, S., Coelho, S., Hosahalli Lakshmaiah, S., & Agrawal, A. (2025). Overview of CoLI-Dravidian 2025: Word-level Code-Mixed Language Identification in Dravidian Languages. In *Forum for Information Retrieval Evaluation (FIRE 2025)*, Varanasi, India.

[14] Hegde, A., Anusha, M. D., Coelho, S., Shashirekha, H. L., & Chakravarthi, B. R. (2022). Corpus Creation for Sentiment Analysis in Code-Mixed Tulu Text. In *Proceedings of the 1st Annual Meeting of the ELRA/ISCA Special Interest Group on Under-Resourced Languages* (pp. 33–40), Marseille, France. European Language Resources Association.