# KannadaText Insights: Building Insights from Scraped Kannada News Data

1st Vivek Vittal Biragoni(211AI041)
*Artificial Intelligence*
*National Institute of Technology, Karnataka*

2nd Raghavendra(211AI029)
*Artificial Intelligence*
*National Institute of Technology, Karnataka*

***Abstract*—In our digital world, languages need a fair representation in tech, especially in NLP. Our project is all about boosting Kannada, an underrepresented language. We creating a diverse Kannada dataset to teach computers to understand it better. Simultaneously, We are exploring what makes Kannada unique.**

***Index Terms*—NLP, Kannada language, Dataset createion**

## I. Introduction

In the ever growing world of digital communication and technology, the representation of diverse languages is integral to the wholesome growth of Natural Language Processing (NLP) technologies. With English and other dominant languages often taking the spotlight, the linguistic richness of regional languages, like Kannada, faces the risk of being overshadowed. With this context, this project plays an important role as we are addressing the most important yet overlooked task in any of the NLP tasks(whole AI too) i.e, to have a dataset for kannada.

The primary objectives of this project are to two-fold: one is to construct a comprehensive Kannada dataset, and the other is dive into the linguistic beauty of Kannada and also to define the uniqueness of the language. The need for this was as there is lesser represetation of kannada in digital format, it gets extremely difficult to carry on with any NLP tasks and the language keeps lagging behind as it never sees the real growth thats happeing with the NLP tech.

After the dataset creation to create word2vec embeddings which would capture the naunces and dialects very well as we are training on a huge dataset. These embeddings can be used for various purposes. We also in future extend it to cover other embeddings like glove, fastext and more.

In the larger context, as widely spoken languages like English exert their global influence, there is a potential risk of regional languages, like Kannada, gradually diminishing in significance. The prevalence of technologies centered around the English language emphasizes the need to empower regional languages. This would result in preserving their unique cultural nuances and ensuring their active participation in the digital world.

## II. Methodology

### A. Objective

The primary objective of this study is to create a substantially large dataset for Kannada by extracting text content from various news websites. To achieve this, an automated web scraping approach using Selenium and Python was built.

### B. Data Collection Methodology

This section outlines the detailed methodology for collecting data to build a comprehensive Kannada dataset.

*1) Seed URLs:* To curate a diverse and comprehensive dataset for Kannada, a set of seed URLs from reputable Kannada news websites was carefully selected. Each seed URL corresponds to a news website, and specific HTML elements on these pages were targeted for data extraction as well as the URL extraction to loop over to get new text if we comacross any not yet reached URLs.

- **URL:** The web address of the news website.
- **URL Selector:** The CSS selector covering the whole presently reached newspaper webpage to get any available new not-yet-touched URls.
- **Text Selector:** The CSS selector pinpointing the specific text content within the main content area.

### C. Web Scraping Implementation

This subsection details the implementation of web scraping using Selenium and Python.

*1) Selenium Setup:* The Selenium WebDriver, coupled with Python, was utilized for automated web browsing. The scraping script navigates through the seed URLs, extracting data and any unvisited URLs from the page for further linguistic analysis.

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import
    WebDriverWait
from selenium.webdriver.support import
    expected_conditions as EC
```

*2) Scraping Function:* The `scrape_links_and_text` function is designed for automated web scraping, extracting both links and text content from a given seed URL. Here is an in-depth overview of its key components:

- **Web Driver Initialization:** The function begins by initializing a Chrome web driver using Selenium, providing a browser environment for automated interactions.
- **Link Extraction:** Using JavaScript execution in the browser, the function retrieves all links on the seed URL page and eliminates duplicates to create a(set) list of unique links.
- **Text Extraction:** For each link that starts with the provided seed URL, the function navigates to the link and waits for the presence of the specified CSS selector (`text_css_selector`). Upon locating the element, it extracts the text content and appends it to the 'extracted_text.txt' file.
- **Error Handling:** The function also has error handling to manage potential exceptions during the scraping process. If an error occurs, it is caught, and a descriptive message is printed to the console.
- **Recursive Link Crawling:** The function further explores the identified links on the current page, ensures a thorough exploration of the website by adding new links to the list of all links.
- **Web Driver Termination:** Finally, the web driver is closed to conclude the scraping process.
- **Large Dataset Generation:** One of the distinguishing features of this function is its efficiency in generating a large-sized dataset. By systematically exploring the links within the provided seed URL and recursively crawling through associated links, the function accumulates a diverse range of text content. This capability is essential for creating a substantial and representative dataset for Kannada language analysis.

This detailed approach ensures comprehensive data collection while handling potential challenges during the web scraping operation.

### *D. Data Preprocessing*

Following data collection, a series of preprocessing steps are applied to ensure the quality and relevance of the scraped text data:

- **Merge Small Lines:** Identify and merge short lines of text for more coherent sentences.
- **Remove Longer Lines in Other Languages:** Detect and remove lines in languages other than Kannada.
- **Handle Special Characters and Symbols:** Address special characters, symbols, or irrelevant formatting.
- **Tokenization:** Tokenize the text into words or smaller units.
- **Lemmatization or Stemming:** Apply lemmatization or stemming to reduce words to their base or root form.
- **Handle Numerical and Special Characters:** Address any numerical values or special characters.

These preprocessing steps collectively contribute to the preparation of the dataset for subsequent linguistic analysis.

### *E. Word-embeddings*

*1) Tokenization:* Before moving on to the word embeddings we have to tokenize the words. The tokenization process is carried out as follows:

1) **File Encoding Detection:**
   - The script begins by detecting the encoding of the input dataset file (`kn.txt`).
   - `UniversalDetector` from `chardet` is used for this purpose.
2) **Streaming Tokenization:**
   - The script reads the entire dataset file line by line, tokenizing each line in a streaming fashion.
   - The `indic_tokenize.trivial_tokenize` function is used for Kannada text, providing a rule-based tokenizer tailored for Indic languages.
   - Tokenized lines are immediately written to the output file (`tokenized_data.txt`) in an append mode, avoiding the need to store all tokenized lines in memory.

*2) :* Word2Vec Model Training with Streaming
The Word2Vec model is trained using the streaming approach.

- **Data Streaming:**
  - The script reads tokenized lines from the `tokenized_data.txt` file in a streaming fashion.
- **Training the Model:**
  - The Word2Vec model undergoes incremental training on each tokenized line through data streaming.

- **Model Parameters:**
  - ∗ **Vector Size (200):** Determines the dimensionality of word vectors, indicating the number of features for each word.
  - ∗ **Window Size (10):** Represents the maximum distance between the current and predicted word within a sentence.
  - ∗ **Min Count (5):** Disregards words with a total frequency below this threshold.
  - ∗ **Workers (12):** Specifies the number of CPU cores utilized for training, enhancing efficiency.
  - ∗ **Skip-Gram (1):** Dictates the training algorithm, with a value of 1 denoting the Skip-Gram model.
  - ∗ **Negative Sampling (5):** Sets the number of noise words for negative sampling.
  - ∗ **Negative Sampling Exponent (0.75):** Defines the exponent shaping the negative sampling distribution.
  - ∗ **CBOW Mean (1):** If 0, utilizes the sum of context word vectors. If 1, employs the mean, applicable only when CBOW is used.
  - ∗ **Alpha (0.025):** Represents the initial learning rate for the model.
  - ∗ **Min Alpha (0.0001):** Establishes the minimum learning rate during training, gradually reducing it.
  - ∗ **Sorted Vocabulary (True):** Involves sorting the vocabulary for enhanced efficiency.
  - ∗ **Max Vocabulary Size (10,000,000):** Imposes a limit on RAM during vocabulary building, pruning infrequent words.
  - ∗ **Sample (0.0001):** Specifies the threshold for determining which higher-frequency words are randomly downsampled.

  - The training process efficiently handles large datasets with a focus on avoiding overfitting by using a smaller number of epochs during streaming.

- **Training Summary:**

  - **Vocabulary Size:** 257209
  - **Unique Documents Processed:** 53266064
  - **Total Training Words:** 604876934

## III. Results

### A. Dataset Generation and Data Crawler

We successfully generated a large-scale Kannada dataset using a robust data crawler built with Selenium. This dataset encompasses diverse textual content from scraped Kannada news sources, providing a rich and varied source for analysis.

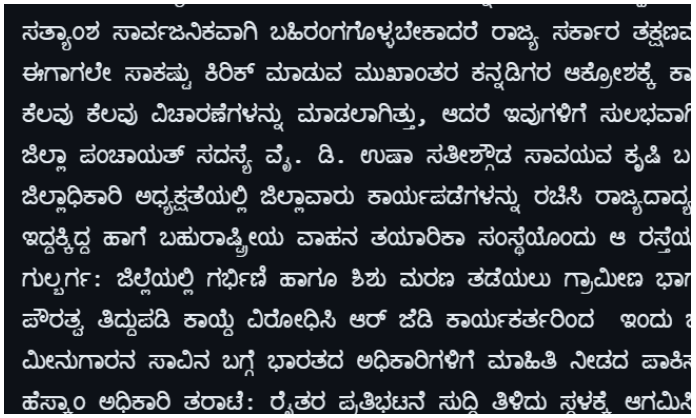


Fig. 1. Sample of the Kannada Dataset

#### 1) A Sample of the Dataset: Fig. 1.:

### B. Word2Vec

We trained a Word2Vec model on the generated Kannada dataset. The model captures semantic relationships between words, enabling us to explore nuanced patterns in Kannada language.

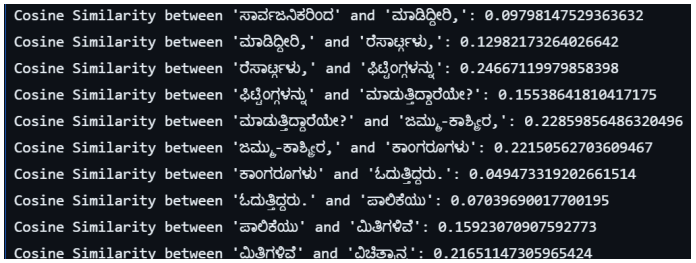


Fig. 2. Cosine similarity of any two words based on our vectors


Fig. 3. king:male::queen:x, top 2 choices for x provided by our vectors

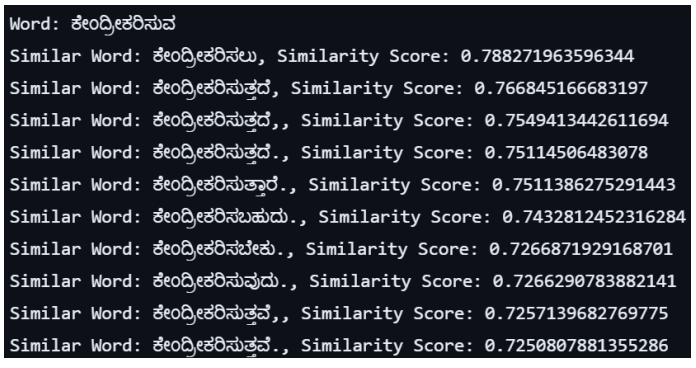


Fig. 4. 10 Words most similar to the vector of the chosen word:

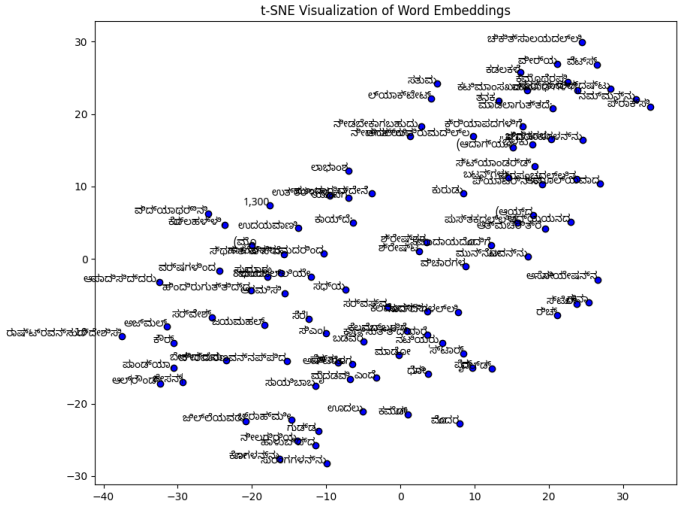


Fig. 5. t-SNE Visualization of Kannada Word Embeddings:

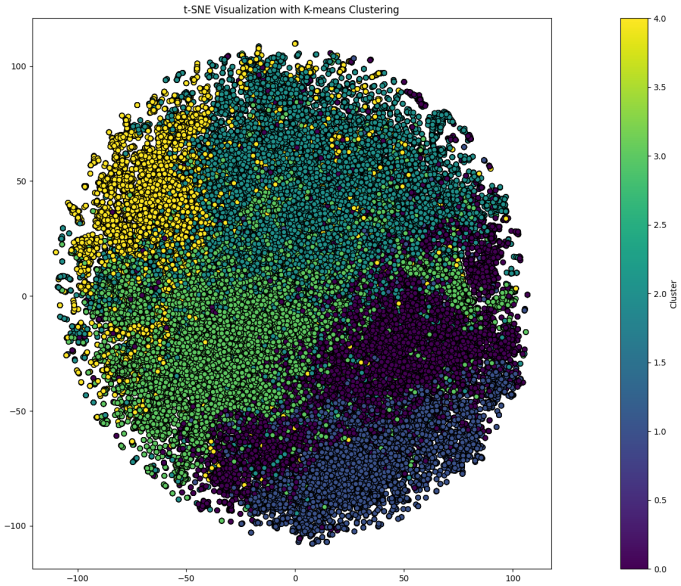


Fig. 6. Word Embedding Clusters Revealed by t-SNE for 1lakh word

## IV. Conclusion

In short, our mix of data, a strong data crawler, and smart word models dig deep into Kannada language. Visuals help us see the language's complexities. This work sets the stage for more language exploration in Kannada's natural processing. Also, our varied models give a base for lots of uses like picking apart meanings, boosting Named Entity Recognition (NER), refining info finding, learning transfer, figuring out topics, blending different modes, making language, creating metrics, trying cross-language tricks, and using in real things like chatbots and feeling checks.

## References

[1] I Gusti Lanang Putra Eka Prismana, Dedy Rahman Prehanto, I Kadek Dwi Nuryana, "The Design and Implementation of Web Crawler Distributed News Domain Detection System," Proceedings of the International Joint Conference on Science and Engineering (IJCSE 2020).

[2] Selenium, "Selenium is a web-scraping library written in Python. We have used it for scraping news websites for content." [Online]. Available: https://www.selenium.dev/

[3] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS.

[4] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. (2017). Enriching Word Vectors with Subword Information. arXiv preprint arXiv:1607.04606.

[5] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[6] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). Roberta: A robustly optimized BERT approach. arXiv preprint arXiv:1907.11692.

[7] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T., Mikolov, J., Grave, E. (2017). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[8] Chen, D., Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 740-750).