Acta Scientific AGRICULTURE (ISSN: 2581-365X)

Volume 5 Issue 5 May 2021

Research Article

Using Natural Language Processing to Translate Plain Text into Pythonic Syntax in Kannada

Vinay Rao[1]*, Sanjana GB[2], Sundar Guntnur[2], Navya Priya N[2], Sanjana Reddy[2] and Pavan KR[2]

[1]Independent Researcher, India

[2]RVCE, Mysore Rd, RV Vidyaniketan, Post, Bengaluru, Karnataka, India

***Corresponding Author:** Vinay Rao, Independent Researcher, Bangalore, India.

Received: March 10, 2021

Published: April 26, 2021

© All rights are reserved by **Vinay Rao., *et al.***

Abstract

Digital evolution has made various services and products available at everyone's fingertips and made human lives easier. It has become necessary for individuals with a passion to be a part of this digital evolution to learn how to write code, which is the basic literacy of the digital age. But writing code has become a privilege for students with prior knowledge of English.

In the context of the evolving field of Agri-tech, individuals and companies are making large strides towards digitising various different aspects of Agriculture. AI is being used actively to solve various problems in the agricultural space. The basic expected literacy here as well, is the ability to write code with the default understanding of English. Rural areas where one of the mainstream occupation for a large part of the population is Agriculture, English language may not be their primary language of choice for written and verbal communications.

With our work, we wish to provide a learning interface that users can employ to first learn the basics of writing code in their native language (Kannada will be in focus in our paper) and in future, the farmers can themselves build and consume tools that help them in their day to day needs with the skill of writing code that they can now acquire without the pre-requisites of knowing English.

The current model can successfully identify and convert conditional statements in the Kannada language into python code. The next effort will be aimed at extending this to recognise loop statements and create a framework for a wide variety of languages.

Keywords: Parts of Speech Tagger (PoS); Programming Languages (PL); Transfer Learning; AWD LSTM; fast.ai; Stemmer; Python; Kannada (Native South Indian Language)

Introduction

Digital technology has made revolutionary changes in human life. It plays an important role in almost every aspect of society. It has helped in the development of education, communication, agriculture, disaster response and many other fields. It helps in the economic growth of a country by improving efficiency.

Coding is the language in this modern digital world. It has become a centre of all business. Learning to code has become essential for anyone to pursue a career in this field. Many languages are available in which coding can be done. A survey shows that of 8500 programming languages available, 2400 were made in the United States, 600 in the United Kingdom, 160 in Canada and 75 in Australia, English is native to these countries. Accordingly, more than one-third of these originated in English-speaking countries. Most of the resources available to learn and understand these languages are also in English [8].

But according to World Language Statistics (SIL International, 2015), English is the 3rd most spoken language in the world, with 5.43% of speakers, behind Chinese and Spanish with 14.4% and 6.15%, respectively. And another survey of the most used Programming Languages' (TIOBE Software BV, 2019) Syntax, Semantics, Standard Library and Runtime System indicates that the most popular are all English based [11].

Even though Non-English-based PLs exist (Miller, Vandome, and McBrewster, 2012), currently the most used have syntax, learning resources, Runtime, and Development Environments that are developed with an English speaking audience in mind [11].

During the month of April of 2015, a survey was conducted on 78 students of the University College of Engineering, Osmania University. The survey was to perceive the importance of native language to understand the source code and the importance of comments in programs to understand the code. Figure 1 and 2 represent the output of the survey [11].

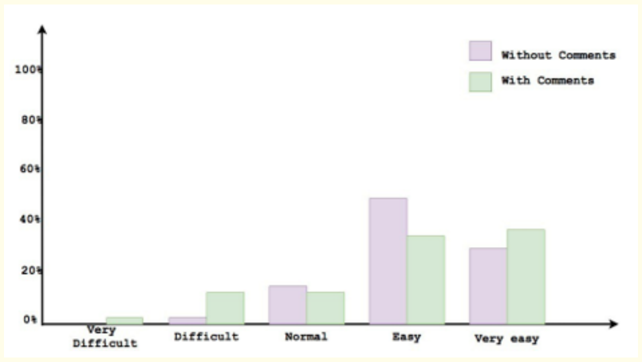


Figure 1: Result of the Survey to check the importance of comments in understanding the source code [11].

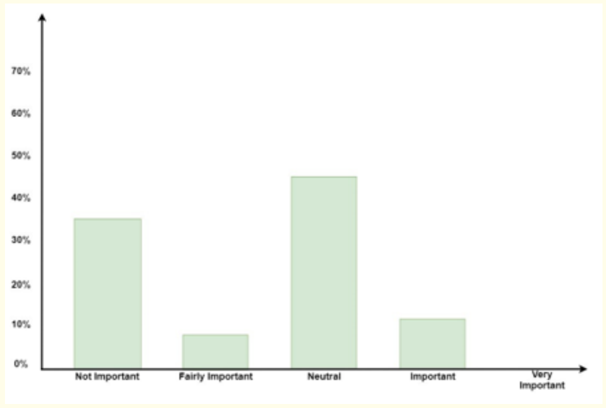


Figure 2: Result of the Survey to check the importance of native language in understanding the source code [11].

From the above survey, it was found that not only do many students find native language important or are neutral towards it, but the source code in the form of comment helps students learn the programming language. This survey highlights the need of the hour which is to make code more accessible to users who may not have English as their primary mode of communication. This survey also throws light on the fact that code when expressed in the form of comments (plain text) does help in narrowing down the logic of what was implemented without needing the user to dig into the syntactic nuances of the coding language.

Therefore, in this paper, we present a learning platform where simple coding questions on basic programming techniques are provided in native language, to which users can provide solutions in their native language; hence allowing students to focus on good logic and problem-solving skills as their launchpad into the world of writing code. Their solution in the native language will then be the input to a model which will convert this statement into python code in their natural language (educational) and English (to execute). This in turn helps the users visualise the translation of logic to code.

This model uses Parts of Speech (PoS) tagging for the native language [2] as the base, along with language semantics words like variables, quantities, conditional words, and actions. A stemmer [15] for the same language is used to find the relational operation to be done in the conditional statement (example, greater than, not greater than).

A large amount of procedural data was obtained from wiki How [26] and was translated to Kannada. This data was then used to develop and evaluate a model that could identify conditional statements in native language plain text and convert it to pythonic code. The flowchart figure 3 represents the platform to be achieved.

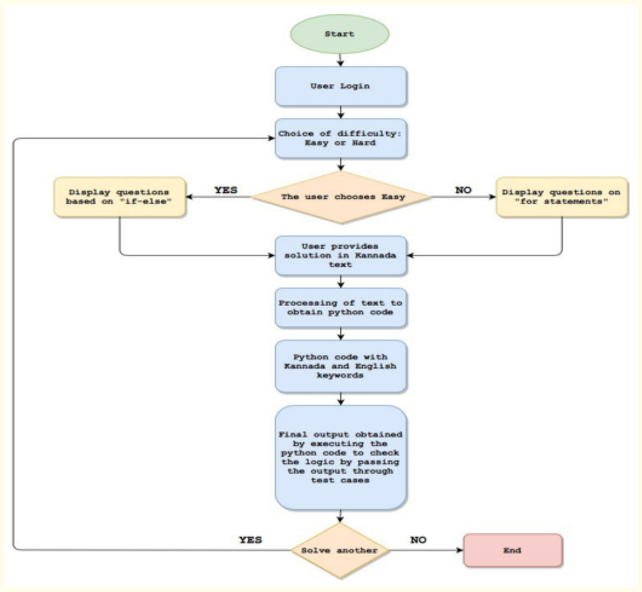
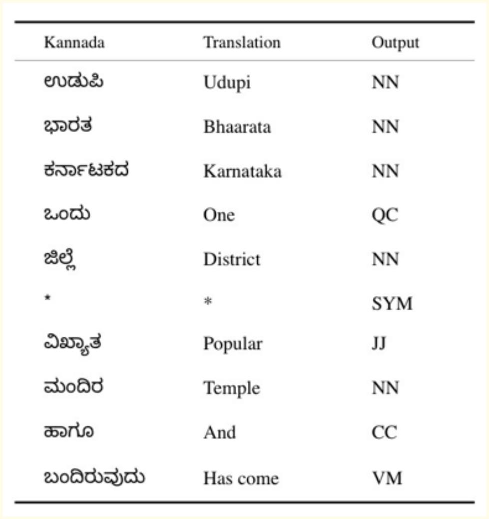

Start
User Login
Choice of difficulty: Easy or Hard
The user chooses Easy
YES
NO
Display questions based on "if-else"
Display questions on "for statements"
User provides solution in Kannada text
Processing of text to obtain python code
Python code with Kannada and English keywords
Final output obtained by executing the python code to check the logic by passing the output through test cases
Solve another
YES
NO
End

Figure 3: Platform aimed to be achieved.

Materials and Methods

This model is based mainly on Parts of Speech tagging. It uses the concept of transfer learning using fast.ai's ULMFit [12]. The RNN model with pre-defined weights requires a dataset for training the model for Parts of Speech tagging. The dataset used for this is called Kasthuri. It contains various words in Kannada vocabulary and the corresponding tag. A snippet from Kasthuri dataset is shown in table 1 [1].

A stemmer [20,23] was also used to extract the root word from conjunctions that usually carry the conditional word. It is a static model that uses dictionaries corresponding to various tenses in Kannada. Each of these dictionaries consist of various suffixes based on which stemming is done.



Kannada	Translation	Output
ಉಡುಪಿ	Udupi	NN
ಭಾರತ	Bhaarata	NN
ಕರ್ನಾಟಕದ	Karnataka	NN
ಒಂದು	One	QC
ಜಿಲ್ಲೆ	District	NN
*	*	SYM
ವಿಖ್ಯಾತ	Popular	JJ
ಮಂದಿರ	Temple	NN
ಹಾಗೂ	And	CC
ಬಂದಿರುವುದು	Has come	VM

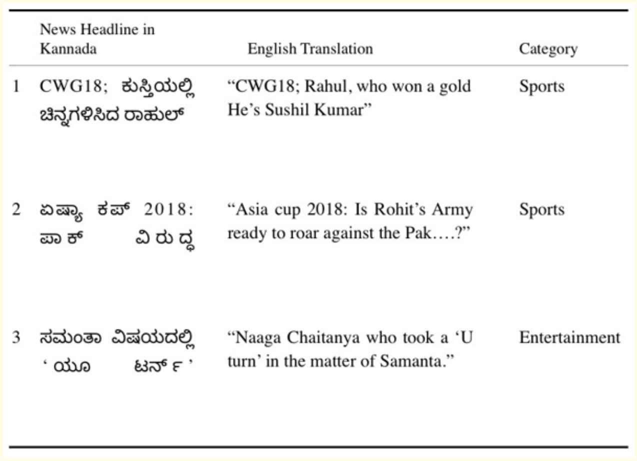
Figure a: Snippet from the Kasturi dataset.

Apart from this, the iNLTK [9] libraries that were used and referenced in this project, uses 6300 news article headlines, which were collected from Kannada news websites and 32000 Wikipedia Articles, which have been cleaned. Table 1 represents the sample headlines present in the iNLTK library and table 1 displays the statistics of various topics in the iNLTK library.

Headlines	Category Split
5114	Entertainment 52%
Unique values	Sports 36%
	Other 12%

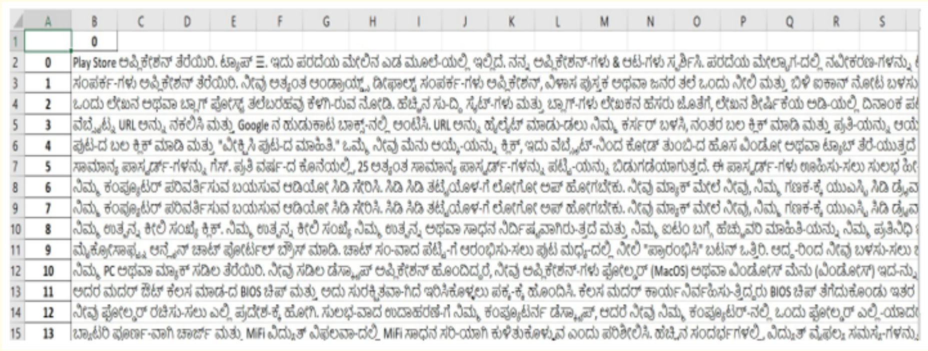
Table 1: Statistics of iNLTK dataset used in this project [9].

To determine the general pattern in conditional statements, the procedural data obtained from wiki How [26] pages were translated to get a dataset of Kannada procedural text, as shown in figure 4.



	News Headline in Kannada	English Translation	Category
1	CWG18; ಕುಸ್ತಿಯಲ್ಲಿ ಚಿನ್ನಗಳಿಸಿದ ರಾಹುಲ್	"CWG18; Rahul, who won a gold He's Sushil Kumar"	Sports
2	ಏಷ್ಯಾ ಕಪ್ 2018: ಪಾಕ್ ವಿರುದ್ಧ	"Asia cup 2018: Is Rohit's Army ready to roar against the Pak….?"	Sports
3	ಸಮಂತಾ ವಿಷಯದಲ್ಲಿ ' ಯೂ ಟರ್ನ್ '	"Naaga Chaitanya who took a 'U turn' in the matter of Samanta."	Entertainment

Figure b: Samples headlines from the dataset.



	A	B
1		0
2	0	Play Store ಅಪ್ಲಿಕೇಶನ್ ತೆರೆಯಿರಿ. ಟ್ಯಾಪ್ ☰. ಇದು ಪರದೆಯ ಮೇಲಿನ ಎಡ ಮೂಲೆ-ಯಲ್ಲಿ ಇಲ್ಲಿದೆ. ನನ್ನ ಅಪ್ಲಿಕೇಶನ್-ಗಳು & ಆಟ-ಗಳು ಸ್ಪರ್ಶಿಸಿ. ಪರದೆಯ ಮೇಲ್ಭಾಗ-ದಲ್ಲಿ ನವೀಕರಣ-ಗಳನ್ನು
3	1	ಸಂಪರ್ಕ-ಗಳು ಅಪ್ಲಿಕೇಶನ್ ತೆರೆಯಿರಿ. ನೀವು ಅತ್ಯಂತ ಆಂಡ್ರಾಯ್ಡ್ ಡೀಫಾಲ್ಟ್ ಸಂಪರ್ಕ-ಗಳು ಅಪ್ಲಿಕೇಶನ್, ವಿಳಾಸ ಪುಸ್ತಕ ಅಥವಾ ಜನರ ತಲೆ ಒಂದು ನೀಲಿ ಮತ್ತು ಬಿಳಿ ಐಕಾನ್ ನೋಟ ಬಳಸು
4	2	ಒಂದು ಲೇಖನ ಅಥವಾ ಬ್ಲಾಗ್ ಪೋಸ್ಟ್ ತಲೆಬರಹವು ಕೆಳಗಿ-ರುವ ನೋಡಿ. ಹೆಚ್ಚಿನ ಸುದ್ದಿ ಸೈಟ್-ಗಳು ಮತ್ತು ಬ್ಲಾಗ್-ಗಳು ಲೇಖಕನ ಹೆಸರು ಜೊತೆಗೆ, ಲೇಖನ ಶೀರ್ಷಿಕೆಯ ಅಡಿ-ಯಲ್ಲಿ ದಿನಾಂಕ ಪ
5	3	ವೆಬ್ಸೈಟ್ನ URL ಅನ್ನು ನಕಲಿಸಿ ಮತ್ತು Google ನ ಹುಡುಕಾಟ ಬಾಕ್ಸ್-ನಲ್ಲಿ ಅಂಟಿಸಿ. URL ಅನ್ನು ಹೈಲೈಟ್ ಮಾಡು-ಡಲು ನಿಮ್ಮ ಕರ್ಸರ್ ಬಳಸಿ, ನಂತರ ಬಲ ಕ್ಲಿಕ್ ಮಾಡಿ ಮತ್ತು ಪ್ರತಿ-ಯನ್ನು ಆಯ
6	4	ಪುಟ-ದ ಬಲ ಕ್ಲಿಕ್ ಮಾಡಿ ಮತ್ತು "ವೀಕ್ಷಿಸಿ ಪುಟ-ದ ಮಾಹಿತಿ." ಒಮ್ಮೆ ನೀವು ಮೆನು ಆಯ್ಕೆ-ಯನ್ನು ಕ್ಲಿಕ್, ಇದು ವೆಬ್ಸೈಟ್-ನಿಂದ ಕೋಡ್ ತುಂಬಿ-ದ ಹೊಸ ವಿಂಡೋ ಅಥವಾ ಟ್ಯಾಬ್ ತೆರೆ-ಯುತ್ತದೆ
7	5	ಸಾಮಾನ್ಯ ಪಾಸ್ವರ್ಡ್-ಗಳನ್ನು ಗೆಸ್. ಪ್ರತಿ ವರ್ಷ-ದ ಕೊನೆಯಲ್ಲಿ, 25 ಅತ್ಯಂತ ಸಾಮಾನ್ಯ ಪಾಸ್ವರ್ಡ್-ಗಳನ್ನು ಪಟ್ಟಿ-ಯನ್ನು ಬಿಡುಗಡೆಯಾಗುತ್ತದೆ. ಈ ಪಾಸ್ವರ್ಡ್-ಗಳು ಊಹಿಸು-ಸಲು ಸುಲಭ ಹೀ
8	6	ನಿಮ್ಮ ಕಂಪ್ಯೂಟರ್ ಪರಿವರ್ತಿಸುವ ಬಯಸುವ ಆಡಿಯೋ ಸಿಡಿ ಸೇರಿಸಿ. ಸಿಡಿ ಸಿಡಿ ತಟ್ಟೆಯೊಳ-ಗೆ ಲೋಗೋ ಅಪ್ ಹೋಗಬೇಕು. ನೀವು ಮ್ಯಾಕ್ ಮೇಲೆ ನೀವು, ನಿಮ್ಮ ಗಣಕ-ಕ್ಕೆ ಯುಎಸ್ಬಿ ಸಿಡಿ ಡ್ರೈವ
9	7	ನಿಮ್ಮ ಕಂಪ್ಯೂಟರ್ ಪರಿವರ್ತಿಸುವ ಬಯಸುವ ಆಡಿಯೋ ಸಿಡಿ ಸೇರಿಸಿ. ಸಿಡಿ ಸಿಡಿ ತಟ್ಟೆಯೊಳ-ಗೆ ಲೋಗೋ ಅಪ್ ಹೋಗಬೇಕು. ನೀವು ಮ್ಯಾಕ್ ಮೇಲೆ ನೀವು, ನಿಮ್ಮ ಗಣಕ-ಕ್ಕೆ ಯುಎಸ್ಬಿ ಸಿಡಿ ಡ್ರೈವ
10	8	ನಿಮ್ಮ ಉತ್ಪನ್ನ ಕೀಲಿ ಸಂಖ್ಯೆ ಕ್ಲಿಕ್. ನಿಮ್ಮ ಉತ್ಪನ್ನ ಕೀಲಿ ಸಂಖ್ಯೆ ನಿಮ್ಮ ಉತ್ಪನ್ನ ಅಥವಾ ಸಾಧನ ನಿರ್ದಿಷ್ಟವಾಗಿರು-ತ್ತದೆ ಮತ್ತು ನಿಮ್ಮ ಐಟಂ ಬಗ್ಗೆ ಹೆಚ್ಚುವರಿ ಮಾಹಿತಿ-ಯನ್ನು ನಿಮ್ಮ ಪ್ರತಿನಿಧಿ
11	9	ಮೈಕ್ರೋಸಾಫ್ಟ್ ಆನ್ಲೈನ್ ಚಾಟ್ ಪೋರ್ಟಲ್ ಬ್ರೌಸ್ ಮಾಡಿ. ಚಾಟ್ ಸಂ-ವಾದ ಪೆಟ್ಟಿ-ಗೆ ಆರಂಭಿಸು-ಸಲು ಪುಟ ಮಧ್ಯ-ದಲ್ಲಿ ನೀಲಿ "ಪ್ರಾರಂಭಿಸಿ" ಬಟನ್ ಒತ್ತಿರಿ. ಆದ್ದ-ರಿಂದ ನೀವು ಬಳಸು-ಸಲು
12	10	ನಿಮ್ಮ PC ಅಥವಾ ಮ್ಯಾಕ್ ಸಡಿಲ ತೆರೆಯಿರಿ. ನೀವು ಸಡಿಲ ಡೆಸ್ಕ್ಟಾಪ್ ಅಪ್ಲಿಕೇಶನ್ ಹೊಂದಿದ್ದರೆ, ನೀವು ಅಪ್ಲಿಕೇಶನ್-ಗಳು ಫೋಲ್ಡರ್ (MacOS) ಅಥವಾ ವಿಂಡೋಸ್ ಮೆನು (ವಿಂಡೋಸ್) ಇದ-ನ್ನು
13	11	ಅದರ ಮದರ್ ಬೋರ್ಡ್ ಕೆಲಸ ಮಾಡ-ದ BIOS ಚಿಪ್ ಮತ್ತು ಅದು ಸುರಕ್ಷಿತವಾ-ಗಿದೆ ಇರಿಸಿಕೊಳ್ಳಲು ಪಕ್ಕ-ಕ್ಕೆ ಹೊಂದಿಸಿ. ಕೆಲಸ ಮದರ್ ಕಾರ್ಯನಿರ್ವಹಿಸು-ತ್ತಿದ್ದರು BIOS ಚಿಪ್ ತೆಗೆದುಕೊಂಡು ಇತರ
14	12	ನೀವು ಫೋಲ್ಡರ್ ರಚಿಸು-ಸಲು ಎಲ್ಲಿ ಪ್ರದೇಶ-ಕ್ಕೆ ಹೋಗಿ. ಸುಲಭ-ವಾದ ಉದಾಹರಣೆ-ಗೆ ನಿಮ್ಮ ಕಂಪ್ಯೂಟರ್ನ ಡೆಸ್ಕ್ಟಾಪ್, ಆದರೆ ನೀವು ನಿಮ್ಮ ಕಂಪ್ಯೂಟರ್-ನಲ್ಲಿ ಒಂದು ಫೋಲ್ಡರ್ ಎಲ್ಲಿ-ಯಾದರ
15	13	ಬ್ಯಾಟರಿ ಪೂರ್ಣ-ವಾಗಿ ಚಾರ್ಜ್ ಮತ್ತು MiFi ವಿದ್ಯುತ್ ವಿಫಲವಾ-ದಲ್ಲಿ MiFi ಸಾಧನ ಸರಿ-ಯಾಗಿ ಕುಳಿತುಕೊಳ್ಳುವ ಎಂದು ಪರಿಶೀಲಿಸಿ. ಹೆಚ್ಚಿನ ಸಂದರ್ಭಗಳಲ್ಲಿ, ವಿದ್ಯುತ್ ವೈಫಲ್ಯ ಸಮಸ್ಯೆ-ಗಳನ್ನು

Figure 4: Platform aimed to be achieved.

Results and Discussion

Previous work and theoretical model

In the space of providing an interface that accepts Natural Language inputs in Indic languages and translates that into executable Python, our research yielded no such previous work. Most of the applications previously developed cater to the space of text translators like Google Translate [25], OmTransliterator [29], or, work in the space of code to code translation like Facebook's work on Transcoding [28] or in the space of building code parsers that translate Kannada script into an executable script [30]. While all of the previous work helped take steps in the right direction to solve the challenge of making code more accessible, we found that our utility had little specific predecessors in terms of translation Natural languages (like Kannada) into executable code.

The model's main aim is to recognise various programming constructs in plain text and convert them into python code. As of now, the model can recognise conditional statements in Kannada text and convert them into python code.

There are various methods of achieving this. There are existing models to recognise conditional statements in English [19], these models could be used on Kannada text by translating the entire text to English first. Google Translate [25] was one of the options. However, given that we wanted to perform the entire task in the chosen native language (Kannada), performing this translation up front was not a viable design choice for us.

The second method is to use natural language processing techniques [4] by translating each Kannada word to English word and applying an English Parts of Speech tagger model [17] on it. But this would not be an efficient method because of the difference in semantics of the two languages. A single word in Kannada may translate to more than two words (containing stop words) in English.

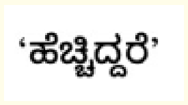


Figure 5: Kannada word for "Greater than".

For example, the Kannada word shown in figure 5, is a single word which translates to "greater than" in English. Hence the word which should be tagged as 'verb' will have two tags. Since there were not any natural language models for Kannada, the first target was to build a good model for Kannada.

Parts of speech tagger for Kannada

PoS tagging on Indian languages, especially Dravidian Languages is a difficult task due to the unavailability of annotated data for these languages. Very little work has been done on Kannada because of the scarcity of good quality annotated data. The recent works in PoS tagging on Kannada have been done with traditional ML techniques like HMM, CRF or SVM [5,14]. PoS tagging was necessary in this case to recognise conditions, actions, variables, and quantities.

The model created was inspired by the backend implementation of the iNLTK [9] libraries to handle use cases in Kannada. The tokenizer and the base model used to perform transfer learning was sourced from the iNLTK [9] codebase. On top of this, a classifier was built using the fast.ai framework [13] that facilitates simple APIs to build a language model and a subsequent classifier model. iNLTK [9] stands for natural language toolkit for Indic languages. It is an open-source Deep Learning library built on top of PyTorch [27] in python which aims to provide out of the box support for various NLP tasks. As of the date of this work, iNLTK [9] library has natural language processing tools for Kannada along with 11 other languages. It consists of tokenizer which has been trained on Kannada Wikipedia articles and Kannada news headlines to learn the general language domain [9] This tokenizer was used for transfer learning on the fast.ai's ULMFit model, shown in figure 6 [13]. Transfer learning refers to the use of a model that has been trained to solve one problem (such as classifying images containing cats) as the basis to solve some other similar problem (such as classifying images containing dogs) [21]. The neural network used for ULMFit's transfer learning is AWD LSTM. AWD LSTM uses drop connect to prevent overfitting of the LSTM [12]. The ULMFit has the following three steps:

- The LM (Language Model) is trained on a general-domain corpus to capture general features of the language in different layers (here iNLTK's tokenizer).
- The full LM is fine-tuned on target task data (here Kasthuri) using discriminative fine-tuning ('Discr') and slanted triangular learning rates (STLR) to learn task-specific features.
- The classifier is fine-tuned on the target task using gradual unfreezing, 'Discr', and STLR to preserve low-level representations and adapt high-level ones [12].

So, the pre-trained model here will be the tokenizer whose last layers will be used for text classification. Figure 7 represents the iNLKTK dataset used for pre-training. The classification process here will be PoS tagging, trained using the Kasthuri dataset, shown in figure c [1].

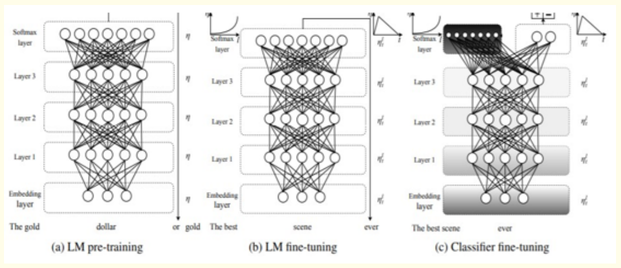
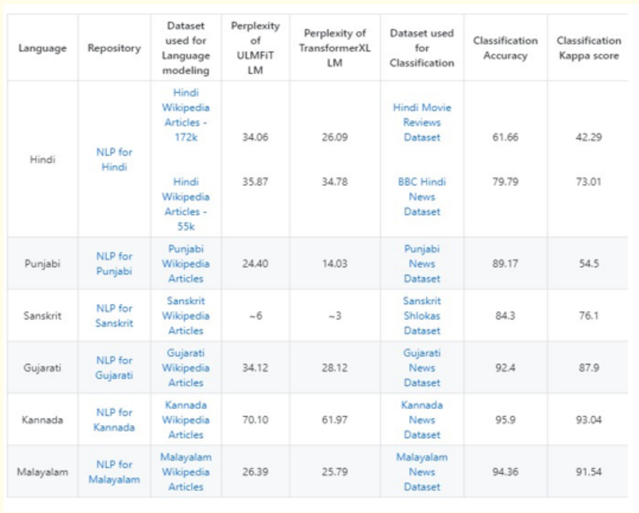
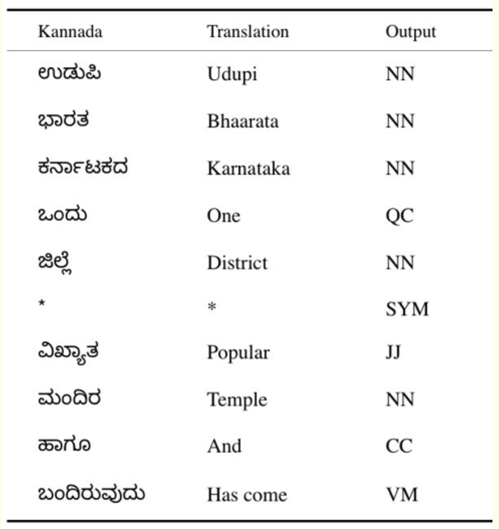


Figure 6: ULMFit on which the POS tagger for Kannada was built [12].



Language	Repository	Dataset used for Language modeling	Perplexity of ULMFiT LM	Perplexity of TransformerXL LM	Dataset used for Classification	Classification Accuracy	Classification Kappa score
Hindi	NLP for Hindi	Hindi Wikipedia Articles - 172k	34.06	26.09	Hindi Movie Reviews Dataset	61.66	42.29
		Hindi Wikipedia Articles - 55k	35.87	34.78	BBC Hindi News Dataset	79.79	73.01
Punjabi	NLP for Punjabi	Punjabi Wikipedia Articles	24.40	14.03	Punjabi News Dataset	89.17	54.5
Sanskrit	NLP for Sanskrit	Sanskrit Wikipedia Articles	~6	~3	Sanskrit Shlokas Dataset	84.3	76.1
Gujarati	NLP for Gujarati	Gujarati Wikipedia Articles	34.12	28.12	Gujarati News Dataset	92.4	87.9
Kannada	NLP for Kannada	Kannada Wikipedia Articles	70.10	61.97	Kannada News Dataset	95.9	93.04
Malayalam	NLP for Malayalam	Malayalam Wikipedia Articles	26.39	25.79	Malayalam News Dataset	94.36	91.54

Figure 7: iNLTK' Kannada tokenizer used for LM pre-training of the ULMFit [9].



Kannada	Translation	Output
ಉಡುಪಿ	Udupi	NN
ಭಾರತ	Bhaarata	NN
ಕರ್ನಾಟಕದ	Karnataka	NN
ಒಂದು	One	QC
ಜಿಲ್ಲೆ	District	NN
*	*	SYM
ವಿಖ್ಯಾತ	Popular	JJ
ಮಂದಿರ	Temple	NN
ಹಾಗೂ	And	CC
ಬಂದಿರುವುದು	Has come	VM

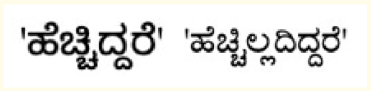
Figure c: Kasthuri dataset used for classifier fine tuning of the ULMFit [1].

Thus, using iNLTK's [9] pre-trained model for Kannada tokenizer and fast.ai's [13] transfer learning method (ULMFit for PoS tagging using the Kasthuri dataset), a Kannada PoS tagger was built.

Stemmer for Kannada

A stemmer was necessary here to recognise relational operations in plain text and determine whether the operation or its opposite had to be performed. The presence of these words indicates that a sentence is a conditional sentence.

For example, words in figure 8, are two Kannada words that stand for 'greater than' and 'not greater than'. Since the root word in Kannada is necessary to determine the operation as 'greater than' and the suffix to determine if it is 'greater than' or 'not greater than', there was the need for a tool to separate root word and the suffix.



'ಹೆಚ್ಚಿದ್ದರೆ' 'ಹೆಚ್ಚಿಲ್ಲದಿದ್ದರೆ'

Figure 8: Kannada words for 'greater than' and 'not greater than.

The stemmer used here was a static model. It splits the word as root word and suffix by comparison of the given words with a set of suffixes. There are 74 arrays in total used in the model. These arrays are based on their tense as well as Parts of Speech. Each of these arrays consists of the suffixes which indicate a conditional statement [20].

Text analysis (Algorithm)

The novelty of our algorithm lies in the observation and translation of language patterns in order to template-ize the occurrence of specific parts of speech and using those extracted parts of speech elements to make decisions on what logical elements they represent. All of the previous work referenced in previous sections target different use cases and therefore present no precedence to compare against.

The Kannada procedural text was passed through the stemmer and PoS tagger. The results obtained were analyzed for a common pattern [3].

The common pattern observed was:

- The first verb in the sentence usually indicated a condition.
- The second verb indicates the end of the action.
- The presence of more than two verbs indicate that it is not a simple conditional statement, but it is either and else, else if or nested condition.

- The first noun in the sentence is the variable.
- The quantities are the terms on which relational operations are performed.
- The word split by the stemmer is the relational operation.

If the root word is same but suffixes are opposite, then it is an else condition. If the root words are different, then it is an else if or nested condition. The presence of words such as the ones in figure 9 and other similar words indicate that it has multiple comparisons in a single conditional statement. The flow of text analysis is shown in figure 10.

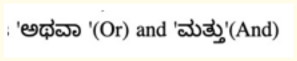

'ಅಥವಾ '(Or) and 'ಮತ್ತು'(And)

Figure 9: Kannada words for “Or” and “And”.

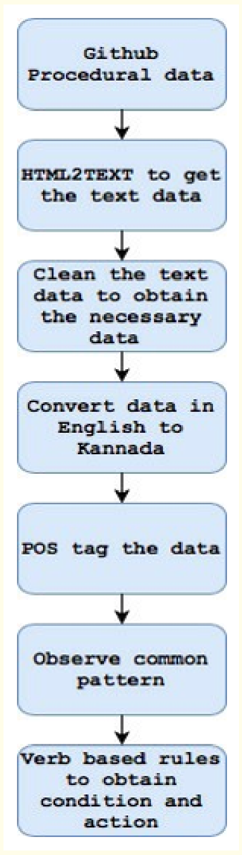

Github Procedural data
HTML2TEXT to get the text data
Clean the text data to obtain the necessary data
Convert data in English to Kannada
POS tag the data
Observe common pattern
Verb based rules to obtain condition and action

Figure 10: Process used to obtain textual dataset in Kannada for analysis and obtaining verb-based rules.

Displaying in Kannada python code

The result obtained from above step is converted to Kannada python code by printing the keywords in Kannada and placing the various extracted features such as variables, quantities, conditions and actions the way they would occur in a python programming construct. For example, code in Kannada would look like figure 11.

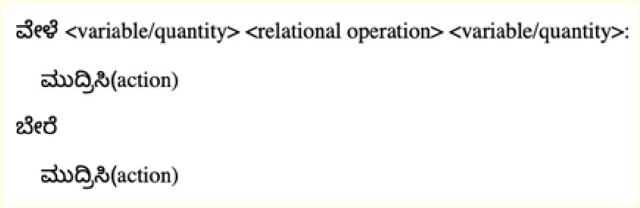
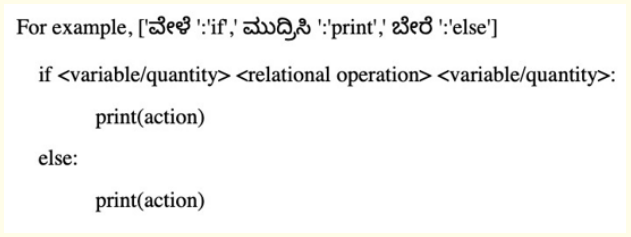

```
ವೇಳೆ <variable/quantity> <relational operation> <variable/quantity>:
    ಮುದ್ರಿಸಿ(action)
ಬೇರೆ
    ಮುದ್ರಿಸಿ(action)
```

Figure 11: Code Snippet in Kannada.

Displaying code in python

The code in Kannada python-format can be converted to English python code by mapping the keywords in Kannada python to English keywords using a simple dictionary. The above code in python will be translated to the following, as shown in figure 12.


For example, ['ವೇಳೆ ':'if',' ಮುದ್ರಿಸಿ ':'print',' ಬೇರೆ ':'else']

```
  if <variable/quantity> <relational operation> <variable/quantity>:
          print(action)
  else:
          print(action)
```

Figure 12: English translation of code with Kannada mapped keywords.

Controlled factors

The controlling factor in this project is the language semantics. In linguistics, semantics is the study of meaning, as inherent at the levels of words, phrases, sentences, and larger units of discourse. These language semantics remain constant in the project and is useful for tokenizing as well as text analysis.

Methods of regression studied

To achieve the mentioned goals, various models were used. There was a trial on StanfordNLP [16] for Kannada to achieve named entity recognition for Kannada and sentiment analysis for positive and negative tags for conditional words as well.

Named entity recognition for Kannada, using StanfordNLP for Kannada, could not be achieved, due to the lack of training dataset. Hence alternative methods had to be figured out. Kasthuri dataset for PoS tagging in Kannada (which was available on GitHub) was used. With this PoS tagger, good accuracy was obtained to complete this task.

Sentiment analysis [18] for negative and positive tags of words to recognise whether the given relational operation must be performed, or its negation was also tried. This seemed like a good method at the beginning. But later it seemed like this task could be achieved using a simple static stemmer model rather than sentiment analysis which seemed like an over-engineered solution.

Hence after studying various methods, it was possible to build a model to achieve these tasks with a PoS tagger and a static stemmer. The workflow of the entire model is represented in figure 13.

Results

Overall, based on examination of the model and the training data, the logic entered by the user in a native language (Kannada) gets tagged to python code. Some of the sentences entered by the user were aimed at outright errors and others at inconsistencies. The rules certainly don't exhaust all the errors and inconsistencies found in the training data, but there are enough covering a number of the most common problem that gives some idea as to whether such taxonomic improvements might noticeably lift the tagging accuracy.

The code works fine for if and if-else conditional statements and gives an accuracy of about 87%. So, the words in the sentence are individually sent to the model which does PoS tagging. From there, the verbs nouns and quantity values are extracted. All the verbs are stemmed and the first verb, when stemmed, gives the condition that the user intends to check against. After which, this verb along with the words following it which specify what needs to be done once the condition is satisfied, are picked, and constructed to the structured python code.

After this, few of the Kannada words are mapped to python keywords, is mapped to "if" figure 14, for example and so on.

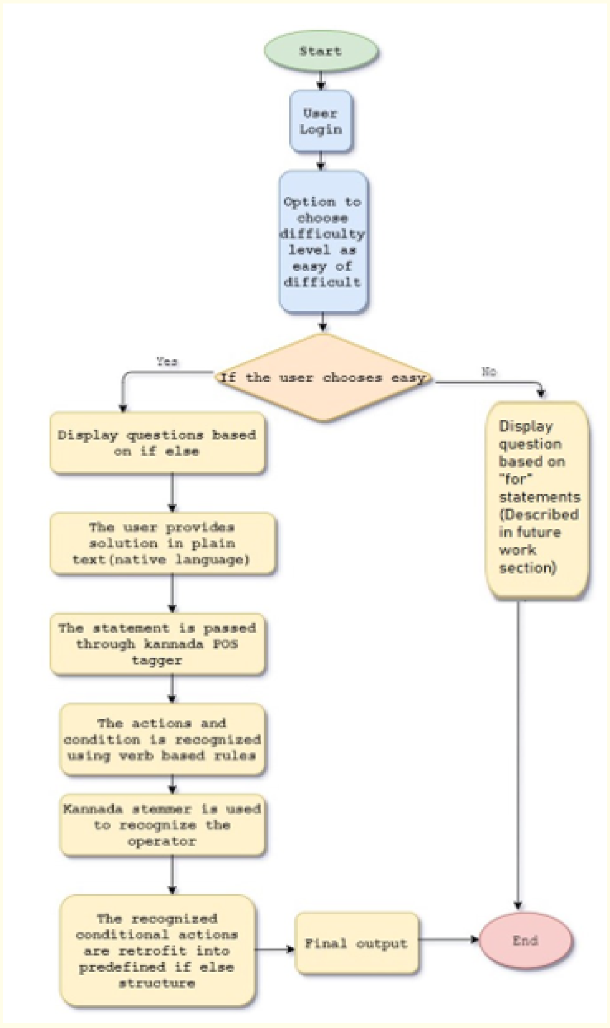
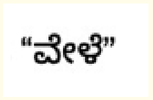

Start
User Login
Option to choose difficoulty level as easy of difficult
If the user chooses easy
Yes
No
Display questions based on if else
The user provides solution in plain text(native language)
The statement is passed through kannada POS tagger
The actions and condition is recognized using verb based rules
Kannada stemmer is used to recognize the operator
The recognized conditional actions are retrofit into predefined if else structure
Final output
End
Display question based on "for" statements (Described in future work section)

Figure 13: Platform Completed can convert any conditional statement in plain text to python code and execute it to obtain final output.

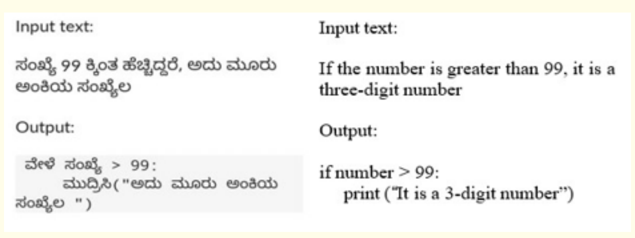


"ವೇಳೆ"

Figure 14: Kannada word for "if".

The code also cleans the text entered by the user to emphasise on the presence of verbs and includes methods to look for opposite words like “not greater than” and a static synonym checker as of now.

Finally, the logic is verified, and the output is given as Kannada python code. If the logic entered is right. The code is consistent and efficient for most of the inputs. The conversion can be evidently seen in the figures figure 15-17.



Input text:

ಸಂಖ್ಯೆ 99 ಕ್ಕಿಂತ ಹೆಚ್ಚಿದ್ದರೆ, ಅದು ಮೂರು ಅಂಕಿಯ ಸಂಖ್ಯೆಲ

Output:

```
 ವೇಳೆ ಸಂಖ್ಯೆ > 99:
     ಮುದ್ರಿಸಿ("ಅದು ಮೂರು ಅಂಕಿಯ
ಸಂಖ್ಯೆಲ ")
```

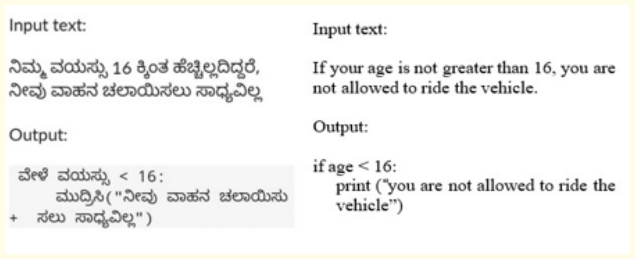
Input text:

If the number is greater than 99, it is a three-digit number

Output:

```
if number > 99:
    print ("It is a 3-digit number")
```

Figure 15: If condition-checks if a given number is a 3-digit number or not. (Right Hand Side-Translation)



Input text:

ನಿಮ್ಮ ವಯಸ್ಸು 16 ಕ್ಕಿಂತ ಹೆಚ್ಚಿಲ್ಲದಿದ್ದರೆ, ನೀವು ವಾಹನ ಚಲಾಯಿಸಲು ಸಾಧ್ಯವಿಲ್ಲ

Output:

```
 ವೇಳೆ ವಯಸ್ಸು < 16:
      ಮುದ್ರಿಸಿ("ನೀವು ವಾಹನ ಚಲಾಯಿಸು
+   ಸಲು ಸಾಧ್ಯವಿಲ್ಲ")
```

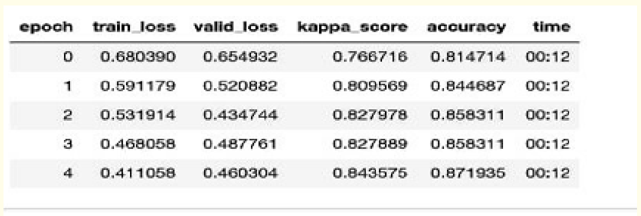
Input text:

If your age is not greater than 16, you are not allowed to ride the vehicle.

Output:

```
if age < 16:
    print ("you are not allowed to ride the
    vehicle")
```

Figure 16: If condition check if the person is eligible to ride/drive or not. (Right Hand Side-Translation) The accuracy for the conditional statements so far is pretty good. It can be evidently seen in Figure 17.



epoch	train_loss	valid_loss	kappa_score	accuracy	time
0	0.680390	0.654932	0.766716	0.814714	00:12
1	0.591179	0.520882	0.809569	0.844687	00:12
2	0.531914	0.434744	0.827978	0.858311	00:12
3	0.468058	0.487761	0.827889	0.858311	00:12
4	0.411058	0.460304	0.843575	0.871935	00:12

Figure 17: Kappa score and Accuracy for the output obtained.

Future Work

The current algorithm is tuned towards helping students understand conditional logic for the “if-then-else” case. Given that each language brings about its semantic nuances with regards to how conditional verbs present themselves in the “greater than/less than” case vs the “is equal to” case, our immediate next goal is to generalise our algorithm to identify all three cases i.e. “less than”, “greater than” and “equal to”. We are also working on “loop” based use-cases. We are currently in the process of exploring NER (Named Entity Recognition) concepts [6,7,10] to extract entities from statements that fit “loop templates” that we are building. We are working towards an algorithm that can consume these entities to solved loop-based problem sets.

Further, we would like to extend this work into other languages and eventually build a platform that exposes APIs to accept formatted language-specific training data that can be used to build a model that works well with the semantic flow of that language. Given the diverse set of languages and the ground work laid by iNLTK [9], we feel that building such a platform will help us reach a larger set of non-English speaking audience who are looking to learn how to code.

We would also like to work towards improving the accuracy of our Parts of Speech tagging model using some of the techniques suggested in other research materials [22,24].

Conclusion

Majority of programming languages and libraries are in English. Non-native English speakers face an obstacle in learning and practicing programming. This language barrier also prevents young students from exploring the world of programming. But a programming language is not just its technical implementation—it is also a human community.

The skill of logical thinking should not be unutilized because of language barriers. Thus, here is the proposal of an idea to make tools and resources to learn programming more accessible to non-native English speakers. This work aims to inspire a future where the knowledge of programming and technical skills is made available to people from all language backgrounds. Hence the non-native English-speaking programmers are also benefitted.

There has been very limited work done to achieve this in Indian languages as well. This work aims to provide a platform to provide knowledge on programming skills to learners who are comfortable with Indian languages and are not well-versed in English.

In the current model, conditional Kannada statements are converted into python code, both with Kannada and English keywords.

As of now, recognition and conversion of conditional statements to code have been successful. Further, this model can be extended to convert Kannada statements, with words involving loops and equal-to cases, into their respective codes. This will help the user understand the basic case of iteration and the involved steps of initialisation, condition and updating.

Further, this model can be successfully extended to more Indian languages and hence it would be more accessible to people from all backgrounds. Currently a static synonym check method is used which will be enhanced to become dynamic.

Eventually, the aim is to create a platform to help students learn coding in their choice of native language, wherein the user is expected to provide a suitable logic for the given problem in plain text. The user-interface will have a text editor where the user will provide their logic. The backend which has the NLP Model as an integral part, will then produce python code in both, native language (as a learning exercise) and in English (to execute) which will be compared with the right solution. The user will be notified about their errors and the right code.

With this work, we wish to make coding more accessible and eliminate the language barrier to writing code. Graphical representation of future work on the platform is represented in figure 18.

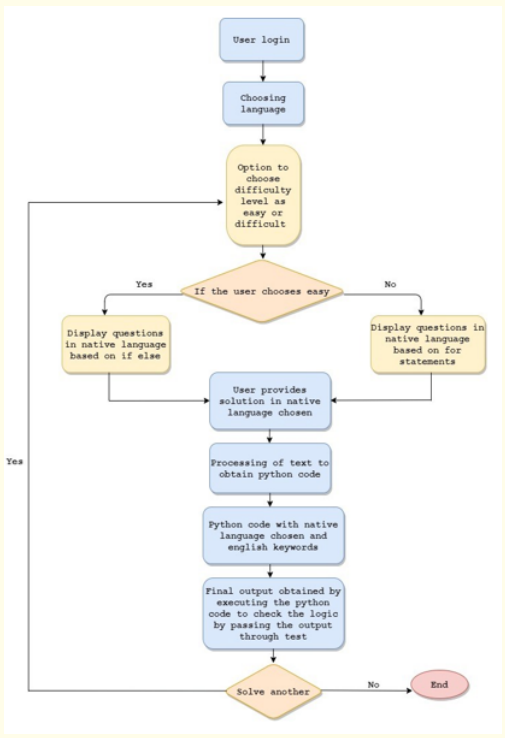

User login
Choosing language
Option to choose difficulty level as easy or difficult
If the user chooses easy
Yes
No
Display questions in native language based on if else
Display questions in native language based on for statements
User provides solution in native language chosen
Processing of text to obtain python code
Python code with native language chosen and english keywords
Final output obtained by executing the python code to check the logic by passing the output through test
Yes
Solve another
No
End

Figure 18: Future work on the Platform to choose any native language to code in.

Acknowledgements

A short acknowledgement section can be written acknowledging the sources regarding sponsorship and financial support. Acknowledging the contributions of other colleagues who are not included in the authorship of this paper should also be added in this section. If there are no acknowledgements, then this section need not be mentioned in the paper.

Conflict of Interest

NA.

Bibliography

1. Adrsh18, Kasthuri dataset.
2. Antony Palesa. "Parts of Speech Tagging for Indian Languages: A Literature Survey". *International Journal of Computer Applications* 34.8 (2011).
3. Baldwin T., *et al.* "Automatic Generation of Conditional Diagnostic Guidelines". *AMIA Annual Symposium Proceedings* 2016 (2017): 295-304.
4. Daniel Jurafsky and James H Martin. "Speech and Language Processing an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition" (2019).
5. Deepamala N and P Ramakanth Kumar. "Kannada Stemmer and Its Effect on Kannada Documents Classification" (2015).
6. Erdem Emekligil., *et al.* "A Bank Information Extraction System Based on Named Entity Recognition with CRFs from Noisy Customer Order Texts in Turkish" (2016).
7. Fang Luo., *et al.* "Product Named Entity Recognition Using Conditional Random Fields" (2011).
8. Frederic P Miller., *et al.* "Non-English-Based Programming Languages" (2012).
9. Gaurav Arora, iNLTK, nlp-for-kannada.
10. Hidayat Ur Rahman., *et al.* "Disease Named Entity Recognition Using Conditional Random Fields".
11. Ivan Ruby., *et al.* "Natural-Language Neutrality in Programming Languages: Bridging the Knowledge Divide in Software Engineering" (2016).
12. Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification" (2018).
13. Jeremy Howard., *et al.* "fastai—A Layered API for Deep Learning" (2020).
14. Ketan Kumar Todi., *et al.* "Building a Kannada POS Tagger Using Machine Learning and Neural Networks Models" (2018).

15. M Thangarasu and R Manavalan. "A Literature Review: Stemming Algorithms for Indian Languages" *International Journal of Computer Trends and Technology (IJCTT)* 4.8 (2013).

16. Manning, Christopher D., *et al.* "The Stanford CoreNLP Natural Language Processing Toolkit". In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2014): 55-60.

17. Raj R., *et al.* "An enhanced Parts of Speech tagger for English sentence". 9 (2016): 6111-6118.

18. Ramanathan Narayanan., *et al.* "Sentiment analysis of conditional sentences". In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1 (EMNLP '09). Association for Computational Linguistics, USA (2009): 180–189.

19. Reinhardt Wenzina and Katharina Kaiser. "Identifying Condition-Action Sentences Using a Heuristic-based Information Extraction Method" (2013).

20. Sahana M. shabdhosk.

21. Sebastian Rudder. "Transfer Learning - Machine Learning's Next Frontier" (2017).

22. Stephen Merity., *et al.* "Regularizing and Optimizing LSTM Language Models" (2017).

23. Suma Bhat. "Statistical Stemming for Kannada" (2013).

24. Vishaal Jatav., *et al.* "Improving Part-of-Speech Tagging for NLP Pipelines" (2017).

25. Yonghui Wu., *et al.* "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation" (2016).

26. "WikiHow for procedural data".

27. Zeming., *et al.* "PyTorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in Neural Information Processing Systems* 32 (2019): 8024-8035.

28. https://arxiv.org/pdf/2006.03511.pdf

29. "OmTransliterator".

30. https://www.researchgate.net/publication/261013439_PraNaMa-_Scripting_Language_in_Kannada

Assets from publication with us

- Prompt Acknowledgement after receiving the article
- Thorough Double blinded peer review
- Rapid Publication
- Issue of Publication Certificate
- High visibility of your Published work

Website: www.actascientific.com/
Submit Article: www.actascientific.com/submission.php
Email us: editor@actascientific.com
Contact us: +91 9182824667