Society for Science and Education United Kingdom
TMLAI Transactions on Machine Learning and Artificial Intelligence
Volume 5, Issue 1
ISSN 2054-7390

Kannada Named Entity Recognition and Classification using Support Vector Machine

[1]S Amarappa, [2]S V Sathyanarayana
[1]*Department of TCE, Jawaharlal Nehru National College of Engineering, Shimoga - 577 204, India;*
[2]*Department of E & C, Jawaharlal Nehru National College of Engineering, Shimoga - 577 204, India;*
amarappas@yahoo.com; svs.tce@gmail.com

ABSTRACT

Named Entity Recognition and Classification (NERC) is a process of identification of proper nouns in the text and classification of those nouns into certain predefined categories like person name, location, organization, date, time etc. Kannada NERC is an essential and challenging work which aims at developing a novel model based on Support Vector Machine. In this paper, tf-idf and POS features are used, which are extracted from a training corpus created manually. Furthermore, the model is trained and tested with different kernels: polynomial, rbf, sigmoid and linear kernels. The details of implementation and performance evaluation are discussed. The experiments are conducted on a training corpus of size 1, 51,440 tokens and test corpus of 7,000, 11,000, 15,000, 20,000, 30,000, 40,000 and 50,000 tokens. It is observed that the model works with an average precision, recall and F1-measure of 87%, 88% and 87.5% respectively for a linear kernel SVM on the test corpus of 7,000 tokens.

Keywords: Natural Language Processing; Hyperplane; Support vectors; Named Entity Recognition; Classification; Support vector machine; Training Corpus; Test Corpus.

1 Introduction

Natural Language Processing (NLP) has two major tasks: Natural Language Understanding (NLU) and Natural Language Generation (NLG) [1]. NLU deals with machine reading comprehension, i.e., the level of understanding of a text or message. NLG is the task of generating natural language from a machine representation system such as a knowledge base. Apart from NLG and NLU, the other tasks to be done in NLP include automatic summarization, Information Extraction (IE), Information Retrieval (IR), Named Entity Recognition (NER) etc.

In NLP, the primary goal of IE and IR is to automatically extract structured information. NERC is a typical subtask of IE [2]. NERC involves processing of structured and unstructured documents and identifying proper names that refer to persons, organizations, locations (cities, countries, rivers, etc.), date, time, etc. The aim of NERC is to automatically extract proper names that are useful to address many problems such as Machine Translation, Information Extraction, Information Retrieval, Question Answering, and Automatic Text Summarization etc., [3].

India has more than 1,652 mother tongues of which 22 are scheduled languages included in the Constitution. Among the 22 scheduled languages, Kannada is one of the major Dravidian languages of

DOI: 10.14738/tmlai.51.2549
Publication Date: 12th February, 2017
URL: http://dx.doi.org/10.14738/tmlai.51.2549

India, spoken predominantly in the state of Karnataka. The Karnataka official language Act 1963 recognized Kannada as its official language. Kannada, whose native speakers are called Kannadigas (Kannadigaru) number roughly 40 million, making it the 33rd most spoken languages in the world (”Census 2001: Languages by state”. censusindia.gov.in. Retrieved on 12 February 2013).

1.1 Kannada language features

The language uses forty-nine phonetic letters, divided into three groups: swaragalu (vowels – thirteen letters); vyanjanagalu (consonants – thirty four letters); and yogavaahakagalu (neither vowel nor consonant - two letters: the anusvara and the visarga), similar to the vowels and consonants of English. The character set is almost identical to that of other Indian languages. This language is inflected with three genders (masculine, feminine, and neutral) and two numbers (singular and plural). The Noun is inflected by various factors such as case, number and gender. It is a free-word order language with rich heritage and large grammar.

1.2 Challenges and Issues specific to Kannada language

Kannada is one of the many Indian languages, presenting a large set of complications. Processing of Kannada language and extraction of named entities on the phrasal semantics basis is challenging because of the reasons:

- Kannada is a highly agglutinating and inflected language.
- Kannada language has no capitalization.
- It has a Brahmi script with high phonetic characteristics that could be utilized by NERC system.
- There is non-availability of large gazetteer, lack of annotated data, lack of standardization and spelling.
- There are a number of frequently used words (common nouns), which can also be used as names.
- These nouns act as adjectives in many contexts and handling these nouns carefully is very much essential. Phrasal semantic analysis of these nouns is interesting.
- As there is lack of annotated data, the whole corpus is annotated by hand. While annotating, care is taken on overlaps among types of Named Entities (NEs). NE overlaps of this kind are carefully tagged based on the phrasal context.

Examples of NE overlaps are:

- Common noun vs. proper noun: ‘surya’ which means sun may be person’s name.
- Organization vs. person name: ‘TaTa’, person name as well as an organization name.
- Organization vs. place name: Mumbai meets Chennai at Bangalore. Here ‘Mumbai’ and ‘Chennai’ are names of playing teams rather than the names of cities.
- Person name vs. place name: The word ‘kashi’ is used as a person name as well as the name of a place.

1.3 Motivation

From the survey carried out in Section 2, it is observed that a lot of work on NERC has been done in English and other foreign languages. NERC work in Indian languages is still in its initial stage. As far as Indian languages are concerned, some works related to NERC are found in Hindi, Bengali, Telugu, Tamil, Oriya, Manipuri, Punjabi, Marathi and Assamese languages. But in Kannada language, NERC work is not yet reported except our former works using Hybrid approach [27], Hidden Markov Model (HMM) [28], Multinomial Nave Bayes (MNB) classifier [29] and Continuous Random Fields (CRF) [30]. The experimental results of all the methods are encouraging; nevertheless the works on NERC in Kannada are to be investigated and implemented with different statistical approaches apart from HMM, MNB and CRF. This has motivated us to take up NERC in Kannada using Support Vector Machine (SVM) classifier as the projected research paper.

1.4 Novelty in this work

With the challenges and issues in Kannada language, we propose the application of Support Vector Machine to resolve the problem of the NERC for Kannada language. We find the work carried out has novelty factors in many respects as mentioned here under:

- The main contribution is that there has been no SVM method available for Kannada language; therefore we ought to deal with the problem from the scratch. This is the first solution of its kind to the problem of NERC using SVM for Kannada language.
- Support Vector Machine model was already used to solve the NERC problem in other languages, but we have to deal with the effort of creating an annotated dataset for the previously neglected language.
- As the annotated Kannada corpus (Unicode) is not available, the whole raw corpus that we have shaped is manually tagged and is checked by local linguistic experts. While annotating we have used fine-grained tags as mentioned in IJCNLP-2008 NERSSEAL shared task data set.
- The work is explained in detail and furthermore, it provides an interesting view over the status of the art with respect to NLP solutions for Indian languages.
- The essence of this work is, tuning up of the SVM idea to the Kannada language NERC.
- The language text is not transliterated (unlike the NERC in other Indian languages) to Roman; instead it is honestly taken from Unicode text files typed by us and trained our model. The test data is also taken from Unicode text files.
- We have used the document classification perception for individual tokens, treating them as independent documents. The features extracted from the training corpus include 'tf-idf' features and parts of speech tags. From these features Support Vector Machine hyperplane is estimated.

This contribution towards Kannada NLP is expected to be a motivation for young researchers and for the readers interested in Information Extraction from natural languages. The application of the results of this work is relevant mostly to research, dealing with Indian languages. So, the work is definitely relevant as it boosts up the scientific developments related to the processing of the Kannada language. Furthermore, the proposed solution was experimentally tested with a variety of test-set sequences and the results are encouraging.

The paper presents in detail the implementation and evaluation of a solution for Named Entity Recognition based on Support Vector Machine for the Kannada language. The results obtained from the proposed model are quite encouraging with an average accuracy of 87% for a linear kernel. The rest of this paper is organized as follows: Section 2 discusses about the details of existing work. Section 3 deals with Support Vector Machine principles which are used for NERC in the paper. The proposed methodology and implementation details are dealt in Section 4. The SVM classifier's evaluation measurements are discussed in Section 5. Finally the results are evaluated and discussed in Section 6 followed by conclusions in Section 7.

2 Existing work

The NLP started way back in the 1940s and from then to 1980s, the NLP systems were based on complex sets of hand-made rules. After 1980s, machine learning algorithms were used in NLP research and recent NLP algorithms are based on statistical machine learning. The term Named Entity was introduced in the sixth Message Understanding Conference (MUC-6) [4]. The different techniques for addressing the NERC problem include: Hidden Markov Models (HMM) (D.Bikeletal, 1997), Decision Trees (S.Sekine, 1998), Maximum-Entropy Models (ME) (A.Borthwick, 1998), Support Vector Machines (SVM) (M.Asahara & Matsumoto, 2003), and Conditional Random Fields (CRF) (A.McCallum & Li, 2003) [31].

A lot of NLP work has been done in English, as there is an enormous amount of data available in it. A voluminous work is done in most of the other European languages, some of the Asian languages like Chinese, Japanese, Korean and other foreign languages like Arabic, etc. NLP research in Indian languages is at the initial stage, as annotated corpus and other lexical resources have started appearing recently. In computational linguistics, Kannada is lagging far behind, compared to other Indian languages. In the following paragraphs, we present a brief survey of research on NERC in Indian languages including Kannada. This is not a comprehensive and thorough survey, but is an indication of current status in NERC research.

Few works on NER in English language are: In [12] the authors have built a CRF based NER system that achieves 91.02% F1-measure on the CoNLL 2003 dataset. An overview of the techniques employed to develop domain specific NER systems is dealt in [13]. In [14] the authors have devised an unsupervised NER by generating Gazetteers and resolving ambiguity.

In [5] the authors have developed an algorithm for rule based NER in Urdu. In [6] the authors carried out a work on Person Name Entity Recognition for Arabic. Reference [7] discusses about SVM based language independent NER. Reference [8] discusses about the first step towards Assamese NER. In [9] the authors have developed a system using CRF approach for NER in Bengali and Hindi. In [10] the authors have developed NER system for Bengali. Reference [11] discussed about Bengali NER using SVM.

In [15] the authors have developed a system for NER in Hindi using Max-Entropy and Transliteration. In [16] the authors have developed Hindi NER by aggregating rule based heuristics and HMM. Reference [17] discusses a composite kernel for NER. In [18] the authors have experimented NER using HMM on Hindi, Urdu and Marathi languages. Reference [19] gives introduction to the CoNLL- 2003 shared task a language-independent NER. Reference [20] discussed about a language independent NER system for Bengali & Hindi using SVM. Reference [21] deals with a model on CRF based NER in Manipuri. Reference [22] deals with SVM based NER for Manipuri. Reference [23] presents the construction of a hybrid, three stages NER for Tamil. In [24] the authors have developed a tourism domain focused NER for Tamil using CRF. Reference [25] describes a Max-Ent, NER system for Telugu. Reference [26] discusses about Telugu NER using language dependent features and rule based approach.

In Kannada language, the only papers available are [27], [28], [29] and [30]. In [27] the authors have carried out Named Entity Recognition Classification and Extraction (NERCE) for Kannada language using hybrid approach, which combines man made rules and Hidden Markov Model (HMM) on a small training corpus of 10,000 tokens and text corpus of 1000 tokens and the experimental results are good with an average F1-measure of 94.85%. In [28] the authors have carried out NERC using Hidden Markov Model

(HMM) on a small training corpus of 10,000 tokens and text corpus of 1000 tokens and the experimental results are encouraging with an average F1-measure of 86%. In [29] we have carried out Kannada NERC based on Multinomial Naïve Bayes (MNB) Classifier and achieved an average F1-measure of 81% on a training corpus of 95,170 tokens and test corpus of 5,000 tokens. In [30] we have carried out Kannada NERC based on Conditional Random Fields (CRF) and achieved an average F1-measure of 82% on a training corpus of 95,127 tokens and test corpus of 5,000 tokens.

3 Support Vector Machine

Although the details of Support Vector Machine (SVM) are well established in the literature, we reiterate the information essential to our research. Support Vector Machine (SVM) is a supervised learning method used for binary classification, regression and outlier's detection. SVM has a simple structure and is derived from statistical learning theory by Vladimir Vapnik and his colleagues in 1992. Given some data points, each belonging to one of two classes and the goal is to decide to which class a new data point belongs. In SVM, a data point is viewed as an n-dimensional vector in n-dimensional space $\mathbf{V} = \mathbf{R}^n$ and we want to know whether we can separate such points with an (n - 1) dimensional hyper plane (canonical plane). There are many hyperplanes that might classify the data but, the best one is with the largest margin. A hyperplane is a subspace of one dimension less than its ambient space. A hyperplane of an n-dimensional space V is a subset with dimension n-1 in V that separates the space into two half spaces. The hyper plane is found by using a subset of training points in the decision function called support vectors, and the margin. To find the margin, two parallel supporting planes are constructed, one on each side of the canonical plane.

3.1 Multi-class SVM

Multiclass classification aims at classifying data points belonging to more than two classes with the assumption that each sample is assigned to one and only one label.

The dominant approach for multiclass classification is to reduce the single multiclass problem into multiple binary classification problems. Common approaches of reducing a multiclass problem into multiple binary classifiers include:

- One-versus-the-rest also known as one-versus-all strategy aims at fitting one classifier per class. If there are n-classes of data points then for each classifier, the class is fitted against all the other n-1 classes and hence it requires n classifier models to be trained. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy and is a fair default choice.
- One-versus-one approach (Knerr et al., 1990) constructs one classifier per pair of classes. At prediction time, the class which received the most votes is selected. If n is the number of data classes, then n * (n - 1) / 2 classifiers are to be constructed and each one trains data from two classes. Since it requires to fit n * (n - 1) / 2 classifiers, this method is usually slower than one-vs-the-rest approach. The basic principle of a binary SVM classifier is derived from the geometrical equation of a straight line y = mx+b, thus defining a linear discriminant function,

$$g(x) = w^T X + b \tag{1}$$

In the Equation (1), w = [w_1, w_2]; w_1 and w_2 are weights of x and y respectively, with 'b' being the intercept to y-axis. The weights w_1 and w_2 may be positive or negative. X is a vector in two-dimensional space. The line $w^T X + b = 0$, is used as a hyperplane in two class classification problems. In SVM, the hyperplane should be found such that it maximizes the margin separating the positive training points from the negative training data points. The Lagrange multiplier of Equation (2) is used to obtain optimized hyperplane, where the term $½||w||^2$ should be minimized, subject to the constraints $Y_i (w^T X + b) \leq 1$.

$$L_P(w, b, \alpha_i) = \frac{1}{2}\,||w||^2 - \sum_{i=1}^{n} \alpha_i (Y_i (w^T + b) - 1) \qquad (2)$$

Such that $\alpha_i \geq 0$. Solving the Equation (2), gives w_1, w_2, b, and α_i. These parameters determine a unique maximal margin solution. The two parallel positive class and negative class supporting planes are constructed, one on each side of the hyperplane by the constraints:

$$w^T X + b = +1 \qquad (3)$$

$$w^T X + b = -1 \qquad (4)$$

4 Proposed work and Methodology

The main aim of this work is to develop a supervised statistical machine learning NERC system for Kannada language based on SVM. NERC involves identification of proper names in texts, and classification of those names into a set of pre-defined categories of interest such as: person names (names of people), organization names (companies, government organizations, committees, etc.), location names (cities, countries etc.), and miscellaneous names (date, time, number, percentage, monetary expressions, number expressions and measurement expressions). The functional block diagram of the proposed system is as shown in Figure 1.

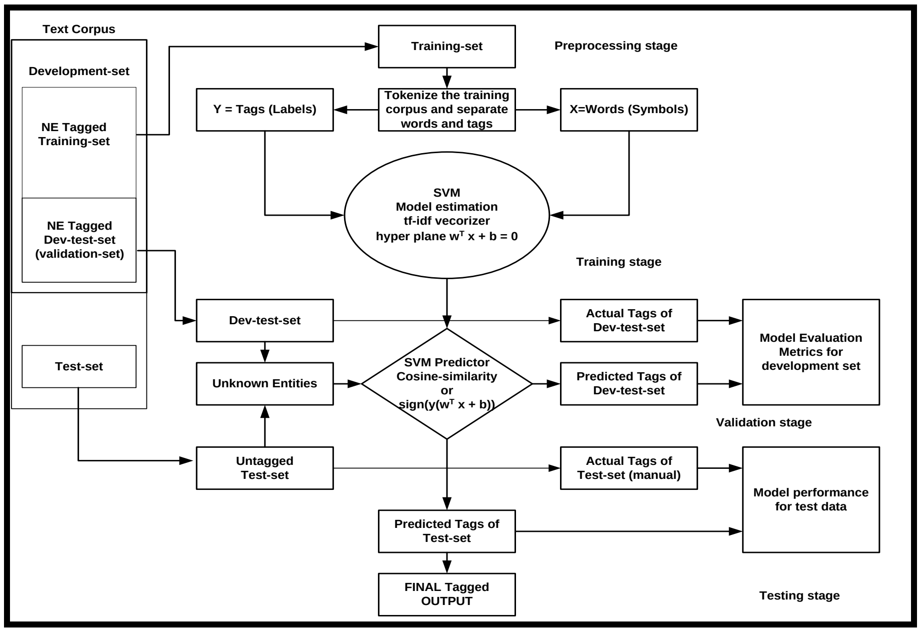

Text Corpus
Development-set
NE Tagged Training-set
NE Tagged Dev-test-set (validation-set)
Test-set
Training-set
Preprocessing stage
Y = Tags (Labels)
Tokenize the training corpus and separate words and tags
X=Words (Symbols)
SVM Model estimation tf-idf vecorizer hyper plane wᵀ x + b = 0
Training stage
Dev-test-set
Unknown Entities
SVM Predictor Cosine-similarity or sign(y(wᵀ x + b))
Actual Tags of Dev-test-set
Predicted Tags of Dev-test-set
Model Evaluation Metrics for development set
Validation stage
Untagged Test-set
Actual Tags of Test-set (manual)
Model performance for test data
Predicted Tags of Test-set
FINAL Tagged OUTPUT
Testing stage

Figure 1: SVM model for Kannada NERC

This Section deals with the design and development of NERC system based on the SVM model. We resent the details of the methodology, design, and development of the proposed system. The machine learning used in the work is fully supervised SVM. The features extracted from the training corpus include POS tag features, term frequency features and inverse document frequency features. The model is trained and tested with different kernels: polynomial, rbf, sigmoid and linear kernels.

4.1 Corpus creation and usage

Kannada NERC is very hard without tagged corpus and hence we manually tagged about 150K Kannada words. This Kannada corpus is used to build the NERC Model. The manually tagged corpus includes: Part of EMILLE (Enabling Minority Language Engineering) corpus [http://www.ciil.org/Schemes.aspx (Linguistic Data Consortium for Indian Languages (LDCIL))], a part of the corpus taken from web articles and part of the corpus extracted from Kannada books. The entire corpus is tagged based on the phrasal semantics taking the text context into consideration. The whole corpus is divided into two sets: Development-set and Test-set as shown in Figure 1. First, select the Development-set and then subdivide it into the Training-set and development test set (Dev-test- set). The Training-set is used to train the model and the Dev-test-set is used to perform error analysis. The Test-set serves for the final evaluation of the system. The machine learning used in the work is fully supervised SVM.

The data set is annotated by four annotators with a common agreement. Tag set is chosen exactly similar to that of tag set used in IJCNLP-2008 NERSSEAL shared task data set. While annotating the corpus manually, Named Entity (NE) overlaps are carefully tagged based on the phrasal context. Examples of NE overlaps are mentioned in Section 1.2. The corpus created is having 70% to 80% of distinctive words of Kannada language.

For a binary SVM classifier the input training-set consists of 'N' number of data points in the form **(X_i, y_i)** where, $X_i \in (X_1, X_2 \dots X_N)$ and $y_i \in (+1, -1)$. Moreover X_i is a point in two dimensional vector space **$(X_i \in R^2)$** and represents the contextual information of the tagged word.

In this paper we have used thirteen Named Entities (NEs) with twenty two tags as indicated in Table 1. A non-named entity is tagged as 'NONE' with label '22'. Person name has 4 tags, where the tag NEP is used for person names having only one word in it. If person name is of two words, first word is tagged with NEPB and second word with NEPE and further if person name is of more than two words, first word is tagged with NEPB, last word is tagged with NEPE and intermediate words are tagged with NEPI. The same rules are followed for locations and organizations also (Table 1).

The sample of training corpus is: Training-set X = [(amar, NEP), (jnnce, NEO), (shimoga, NEL), (shimoga, NEL), (jnnce, NEO), (Davanagere, NEL), (shimoga, NEL), (pesitm, NEO), (sathyanarayana, NEP)].

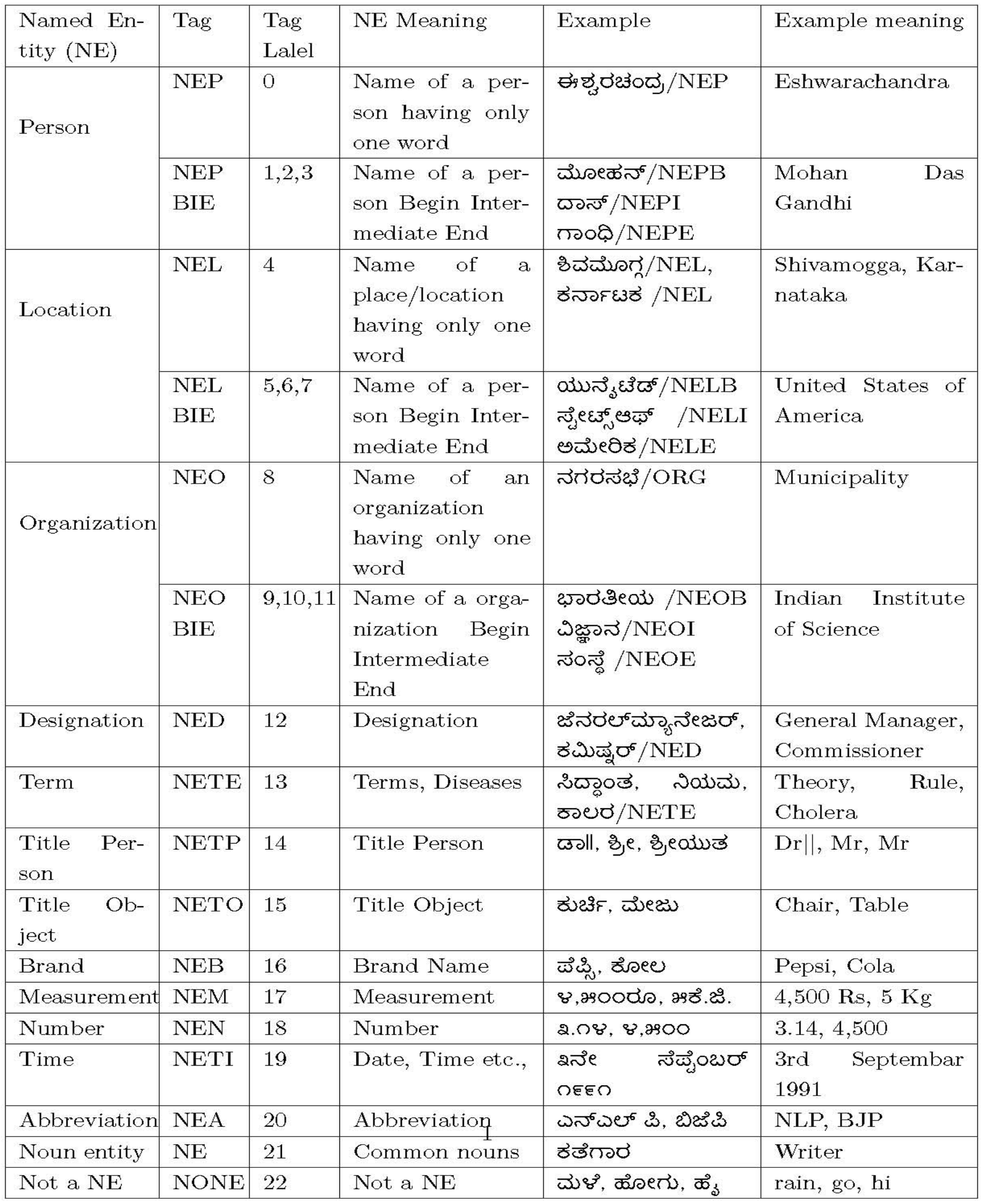
4.2 Pre-processing stage

The tagged training text corpus is tokenized into words (symblols) and tags (states / classes). The separated words are X = [w1, w2, w3 … wN] and separated tags are $Y = [y_1, y_2, y_3 \dots y_N]$. Each tag in Y is assigned a number called label, i.e., $y_i \in (0, 1, 2 \dots 22)$ (or $y_i \in (c_0, c_1 \dots c_{22})$) as given by:

['NEP:0', 'NEPB:1', 'NEPI:2', 'NEPE:3', 'NEL:4', 'NELB:5', 'NELI:6', 'NELE:7', 'NEO:8', 'NEOB:9', 'NEOI:10', 'NEOE:11', 'NED:12', 'NETE:13', 'NETP:14', 'NETO:15', 'NEB:16', 'NEM:17', 'NEN:18', 'NETI:19', 'NEA:20', 'NE:21', 'NONE:22'].

For the sample training-set mentioned in Step 1, tokenize and separate words and tags/labels: Separated words: X = [amar, jnnce, shimoga, shimoga, jnnce, davanagere, shimoga, pesitm, sathyanarayana] and Separated tags/labels: Y = [NEP:0, NEO:8, NEL:4, NEL:4, NEO:8, NEL:4, NEL:4, NEO:8, NEL:4].

Table 1. Named Entity Tag set



Named Entity (NE)	Tag	Tag Lalel	NE Meaning	Example	Example meaning
Person	NEP	0	Name of a person having only one word	ಈಶ್ವರಚಂದ್ರ/NEP	Eshwarachandra
	NEP BIE	1,2,3	Name of a person Begin Intermediate End	ಮೋಹನ್/NEPB ದಾಸ್/NEPI ಗಾಂಧಿ/NEPE	Mohan Das Gandhi
Location	NEL	4	Name of a place/location having only one word	ಶಿವಮೊಗ್ಗ/NEL, ಕರ್ನಾಟಕ /NEL	Shivamogga, Karnataka
	NEL BIE	5,6,7	Name of a person Begin Intermediate End	ಯುನೈಟೆಡ್/NELB ಸ್ಟೇಟ್ಸ್ಆಫ್ /NELI ಅಮೇರಿಕ/NELE	United States of America
Organization	NEO	8	Name of an organization having only one word	ನಗರಸಭೆ/ORG	Municipality
	NEO BIE	9,10,11	Name of a organization Begin Intermediate End	ಭಾರತೀಯ /NEOB ವಿಜ್ಞಾನ/NEOI ಸಂಸ್ಥೆ /NEOE	Indian Institute of Science
Designation	NED	12	Designation	ಜೆನರಲ್ಮ್ಯಾನೇಜರ್, ಕಮಿಷ್ನರ್/NED	General Manager, Commissioner
Term	NETE	13	Terms, Diseases	ಸಿದ್ಧಾಂತ, ನಿಯಮ, ಕಾಲರ/NETE	Theory, Rule, Cholera
Title Person	NETP	14	Title Person	ಡಾ॥, ಶ್ರೀ, ಶ್ರೀಯುತ	Dr||, Mr, Mr
Title Object	NETO	15	Title Object	ಕುರ್ಚಿ, ಮೇಜು	Chair, Table
Brand	NEB	16	Brand Name	ಪೆಪ್ಸಿ, ಕೋಲ	Pepsi, Cola
Measurement	NEM	17	Measurement	೪,೫೦೦ರೂ, ೫ಕೆ.ಜಿ.	4,500 Rs, 5 Kg
Number	NEN	18	Number	೩.೧೪, ೪,೫೦೦	3.14, 4,500
Time	NETI	19	Date, Time etc.,	೩ನೇ ಸೆಪ್ಟೆಂಬರ್ ೧೯೯೧	3rd Septembar 1991
Abbreviation	NEA	20	Abbreviation	ಎನ್ಎಲ್ ಪಿ, ಬಿಜೆಪಿ	NLP, BJP
Noun entity	NE	21	Common nouns	ಕತೆಗಾರ	Writer
Not a NE	NONE	22	Not a NE	ಮಳೆ, ಹೋಗು, ಹೈ	rain, go, hi

We are using the concept of document classification where, each input word is treated as a document and tag as its class: for the above example we have the details as mentioned in Table 2.

Table 2. Each word is a document

Document No.	word (x)	tag (y)
D1	amar	NEP

D2	jnnce	NEO
D3	shimoga	NEL
D4	shimoga	NEL
D5	jnnce	NEO
D6	davanagere	NEL
D7	shimoga	NEL
D8	pesitm	NEO
D9	sathyanrayana	NEP

4.3 Training stage

The various steps of training the model are as given below:

3.1 Input to the training stage are X and Y

3.2 The model finds important words (vocabulary features) by removing repeated words and stop words from X. The model also finds unique tags/ labels from Y. The vocabulary feature words are W = [w_1, w_2, w_3 … wn] and unique tags are Y = [y_1, y_2, y_3 … y_k].

For the sample training-set mentioned in Step 1, the important words (vocabulary features) by removing repeated words and stop words are: W = [w_1: amar, w_2: jnnce, w_3: shimoga, w_4: davanagere, w_5: pesitm, w_6: sathyanarayana] and the unique tags/labels are Y = [NEP: 0, NEO: 8, NEL: 4].

3.3 Find raw count of each vocabulary word of W in Training-set, i.e., term frequencies tf.

3.4 The term-frequency is a measure of how many times a particular term of W, is present in the document of Training-set X=[x_1, x_2 … x_N] (or D= [d_1, d_2 … d_N]). In our model, each word of X is treated as a document (word x_1 = document D_1). The term-frequency is defined as a counting function and is given in Equation (5).

$$\mathrm{tf}(t,d) = \sum_{x \in d} \mathrm{fr}(x,t) \tag{5}$$

Where fr(x, t) is a simple function, defined by Equation (6).

$$\mathrm{fr}(x,t) = \begin{cases} 1, & \text{if } x = t \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The $\mathrm{tf}(t,d)$ returns count of t in document d. The $\mathrm{tf}(t,d)$ in matrix form is denoted by Equation (7).

$$M_{|D|\times F} = (M_{\mathrm{train}}) \tag{7}$$

3.5 Find inverse document frequency $\mathrm{idf}(t)$ of training corpus defined by the function $P(t|d) = \frac{|\{d: t \in d\}|}{|D|}$, so idf is define as Equation (8).

$$\begin{aligned} \mathrm{idf} &= -\log P(t|d) \\ &= \log \frac{1}{P(t|d)} \\ \mathrm{idf}(t) &= \ln\left(\frac{|D|+1}{1+|\{d: t \in d\}|}\right) + 1 \end{aligned} \tag{8}$$

Here $|\{d: t \in d\}|$ is the number of documents where the term t appears; when the term-frequency function satisfies $\mathrm{tf}(t,d) \neq 0$. It should be noted that adding 1 into the formula above avoids zero division.

3.6 Now to find tf-idf use the following steps tf-idf is found using Equation (9).

$$3.6.1.\ \mathrm{tf}-\mathrm{idf} = \ \mathrm{tf}(\mathrm{t},\mathrm{d}) \times \mathrm{idf}(\mathrm{t}) \tag{9}$$

3.6.2. Find idf for each feature present in the feature matrix with the term frequency and idf weights can be represented by a vector as given by Equation (10).

$$idf^{\rightarrow}{}_{train} = [idf(t_1),\ idf(t_2) \ldots idf(t_k)] \tag{10}$$

3.6.3. tf-idf matrix of training set in un-normalized form is found by:

Now the tf matrix, $\mathrm{M}_{|\mathrm{D}|\times\mathrm{F}} = \mathrm{M}_{\mathrm{train}}$ of Equation (7) and the idf matrix $\mathrm{idf}^{\rightarrow}{}_{\mathrm{train}}$ of Equation (10) are multiplied to calculate the tf-idf weights.

3.6.4. And then multiply $\mathrm{M}_{\mathrm{idf}}$ to the term frequency matrix, so the final result can be defined as Equation (11).

$$[M_{ft-idf}]_{ixk} = [M_{train}]_{ixk} \times [M_{idf}]_{kxk} \tag{11}$$

3.6.5. tf-idf matrix of Training-set in normalized form is given in Equation (12).

$$M_{ft-idf} = \frac{M_{ft-idf}}{\|M_{ft-idf}\|_2} \tag{12}$$

tf–idf vectors are the actual trained parameters/features of the SVM model (Scikitlearn version 0.14 documentation). The tf–idf vectors and POS tags are the main features that are used to determine the hyperplane weight vectors w_1, w_2 and the intercept ‘*b*’.

As already mentioned in Table 1, we have used 13 named entities with 22 classes and a non-named entity is assigned a tag ’NONE’, and hence 23 classifiers are trained: [SVM_{m0}, SVM_{m1} … SVM_{m21}, SVM_{m22}].

The SVM_{m0} is trained such that it assigns positive value (≥ +1) for class c_0 and negative value (≤ -1) for remaining classes. In general SVM_{mi} is trained to give positive result for class c_i and negative result for rest of the classes.

To predict the class of an unknown (test) feature vector ‘p’, the classifier uses the separating hyperplane $w_i^T p + b_i = 0$. If $w_i^T p + b_i \geq +1$, then feature vector ‘p’ belongs to class c_i. Else if $w_i^T p + b_i \leq -1$, then feature vector ‘p’ does not belongs to class c_i.

4.4 Validation stage

A fold of the tagged training corpus is reserved as Dev-test-set and multiple evaluations are performed on various Dev-test-sets. The scores thus obtained from those evaluations are combined to get the average score. A fold of the annotated training data is taken from the Development-set as Dev-test-set and the following computations are performed:

a) Pre-processing and tf-idf computations are done for Dev-test-set as explained in Steps 2 and 3.

b) The tf-idf vector of each sample of Dev-test-set is given to classifier SVM_{m0}. The classifier SVM_{m0} assigns a positive value (≥ +1) if the sample belongs to class c_0. If the sample doesn’t belong to class c_0 it assigns a negative value (≤ -1), and then sample is fed to SVM_{m1}. The classifier SVM_{m1} says whether the sample belongs to class c1 or not. If not the sample is fed to next classifier, and this process is continued till the sample is classified for a right class.

Figure 2 shows the tree of SVM decoding. Here values ≥ +1 are normalized to +1 and values ≤ -1 are normalized to -1.

c) From the actual class labels and predicted class labels of Dev-test set, find precision, recall and F1-measure. Repeat this process on all folds of Dev-test-set and calculate average F1-measure thus validating the model.

4.5 Testing stage

Test-set is taken from the corpus set. The computations are performed similar as in validation stage:

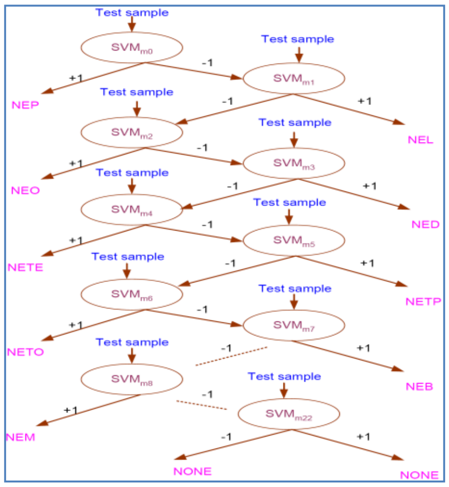


Figure 2. Multi-class SVM for named entity classification

The following algorithm gives the implementation procedure of the SVM Model:

Algorithm:

```
1.    Reading the tagged corpus from the directory and dividing into 10-folds
      corpus ← read tagged corpus
      corpus size ← count of tokens in whole corpus
      folds ← divide the corpus size into ten equal folds
2.    10-fold cross validation of the model
      tag_set = [NEP, NEL, NEO, NED, NETE, NETP, NETO, NEB, NEM, NEN, NETI, NEA, NE, NEPB,
      NEPI, NEPE, NELB, NELI, NELE, NEOB, NEOI, NEOE, NONE]
      tag_set_labels = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]
      Begin
      (i) Preprocessing
            development test set ←reserve ith fold of training
            corpus training set ← take the remaining nine folds as training set
            words ← separate words of training set
            tags ← separate tags of training set
            labels ← assign labels to the tags of training set

      (ii) Feature extraction and training of SVM model

            vectorizer ← TfidfVectorizer
            time0 ← read system time
            training words ← transform words of step (i) into vectors using vectorizer
            classifier ← svm.SVC(kernel='linear')
            (kernels used are: linear, rbf, sigmoid and poly)
            SVM classifier ← input training words and labels to the classifier
            time1 ← read system time
            training time ← time1 – time0
        (iii) Testing with reserved fold of the development test set
            development test set (DTS) ← take reserved ith fold of training corpus
            words of DTS ← separate words of development test set
            actual tags of DTS ← separate tags of development test set
            actual labels of DTS ← assign labels to the tags of development test set
            time0 ← read system time
            test words of DTS ← transform words of DTS  into vectors using vectorizer
            SVM predicted labels of DTS ← SVM classifier receives test words of DTS as input
            time1 ← read system time
            fold test time ← time1 – time0
        (iv) Evaluation metrics
            precision ← precision score from actual labels & SVM predicted labels of DTS
            recall ← recall score from actual labels & SVM predicted labels of DTS
            f1 ← f1 score from actual labels & SVM predicted labels of DTS
            class report ← class report from actual labels & SVM predicted labels of DTS
        End
        Combine the scores of all the ten folds for the cross validation.
3.      Testing of the SVM model
        (i) Train the SVM model for all the 10 folds of training corpus as explained in (ii) of Step 2
        (ii) Testing with Test-set
            test set ← read untagged test corpus from test corpus root directory
            words of test set ← words of test set
            actual tags of test set ← find actual tags of test set (manually)
            actual labels of test set ← assign labels to the tags of test set
            time0 ← read system time
            test words of test set ← transform words of test set into vectors using vectorizer
            SVM predicted labels ← SVM classifier receives test words of test set as input
```

```
time1 ← read system time
test time ← time1 – time0
(iii) Evaluation metrics and classification report are found similar to (iv) of Step 2
```

5 Performance Evaluation Metrics

It is important to know the quality of the SVM machine learning algorithm. Several statistical measurements can be used to estimate the performance of the algorithm. These measurements are collected from a confusion matrix show in Table 3, which contains information about the real and predicted classifications done by the algorithm (https://en.wikipedia.org/wiki/Precision_and_recall).

True positives (TP) - the number of correct predictions that an instance is positive

True negatives (TN) - the number of correct predictions that an instance is negative

False positives (FP) - the number of incorrect predictions that an instance is positive

False negatives (FN) - the number of incorrect predictions that an instance is negative

The aim of the algorithm is to maximize the TP and true negatives TN predictions. The effectiveness of the algorithm is characterized with the recall and precision measurements.

Table 3. Confusion Matrix

		PREDICTED CLASS	
		YES	NO
ACTUAL CLASS	YES	TP	FN
	NO	FP	TN

Recall (R) is the Sensitivity or True Positive Rate (TPR) that measures the ability of the algorithm to find all relevant entities as given by Equation 13. R = Number of correct answers - produced / Total number of possible - correct answers

$$R\ (TPR) = TP/\ (TP + FN) \qquad (13)$$

A high recall score tells that most of the relevant entities were retrieved by the algorithm, while a low recall indicates that the most relevant entities were missed by the algorithm.

Precision (P) measures the ability of the algorithm to retrieve only relevant entities, which is computed using Equation 14.

P = Number of correct answers - produced / Total number of answers – produced

$$P = TP/\ (TP + FP) \qquad (14)$$

A high precision score indicates that most of the retrieved entities are relevant. A low precision means that the algorithm cannot distinguish relevant entities while retrieving all entities.

F1-Measure is the efficiency measure that combines recall and precision together.

F1 -Measure is the traditional F1-measure or balanced F1-score.

$$F1 - Measure\ (F1) = 2PR/\ (P + R) \qquad (15)$$

$$Accuracy = (TP + TN)/\ (TP + FP + TN + FN) \qquad (16)$$

Fallout or False Positive Rate (FPR)

$$FPR = FP/ (FP + TN) \tag{17}$$

Specificity or True Negative Rate (TNR)

$$TNR = TN/ (TN + FP) \tag{18}$$

Miss Rate or False Negative Rate (FNR)

$$FNR = FN/ (FN + TP) \tag{19}$$

$$TPR + FNR = 1 \tag{20}$$

$$TNR + FPR = 1 \tag{21}$$

6 Results and Discussions

The proposed system is designed and implemented as discussed in Section 4. The system is tested using several test cases, containing training corpus of size 1, 51,440 tokens. The test corpus is chosen in such a way that it satisfies the entire phrasal context; which is an inherent feature of Kannada language. It is to be noted that the system achieves an average accuracy of 87% on a test corpus of 7000 tokens with linear kernel. The details of the results obtained are as given below. The system's performance is measured in terms of Precision (P), Recall (R) and F1-measure (F1) as discussed in Section 5. The details of the corpus created in this work are given in Section 4. The nature of input test sequence and output tagged sequence are given in Table 4 and Table 5 respectively. The corpus size and program run time are tabulated in Table 6. Table 7 tabulates the results of 10 fold cross validation where validation fold is of size 15,144 tokens. Table 8 indicates the confusion matrix of the experiment. Table 9 indicates the total count of NE's in the training corpus. Table 10 indicates the final classification results of test-set corpus of size 7000 tokens with linear kernel and Table 11 shows the error analysis.

Table 4. Input test sequence

ವಾಷಿಂಗ್ಟನ್ (ಪಿಟಿಐ): ಮುಂಬರುವ ಲೋಕಸಭಾ ಚುನಾವಣೆ ಬಳಿಕ ನರೇಂದ್ರ ಮೋದಿಯೊಂದಿಗೆ. ಅಮೆರಿಕ ರಾಜತಾಂತ್ರಿಕ ಕೆಲಸ ನಿರ್ವಹಿಸಲು ಸಿದ್ದವಿದ್ದು ಇಲ್ಲಿ ವೀಸಾ ಪ್ರಶ್ನೆಯೇ ಇಲ್ಲ ಎ◌ಂದು ಅಮೆರಿಕ ಸ್ಪಷ್ಟಪಡಿಸಿದೆ. ಮೋದಿ ವಿಶ್ವದ ಅತಿ ದೊಡ್ಡ ಪ್ರಜಾಪ್ರಭುತ್ವ ರಾಷ್ಟ್ರದ ನಾಯಕನಾದರೆ ಅವರ ಜೋತೆ ನಾವು ಕೆಲಸ ಮಾಡುತ್ತೇವೆ ಎ◌ಂದು ಒಬಾಮಾ ಆಡಳಿತದ ಹಿರಿಯ ಅಧಿಕಾರಿಗಳು ತಿಳಿಸಿದ್ದಾರೆ. ಭಾರತೀಯ-ಜನತಾ-ಪಕ್ಷ ಮೋದಿ ಅವರನ್ನು ಪ್ರಧಾನಿ ಅಭ್ಯರ್ಥಿ ಎ◌ಂದು ಘೋಷಿಸಿದೆ. ಒಂದು ವೇಳೆ ಆ ಪಕ್ಷ ಅಧಿಕಾರಕ್ಕೆ ಬಂದರೆ ಮೋದಿಗೆ ಗೌರವ ನೀಡಬೇಕು. ಹಾಗಾಗಿ ಯಾವುದೇ ವೀಸಾ ಸಮಸ್ಯೆ ಉದ್ಭವಿಸುವುದಿಲ್ಲ ಎ◌ಂದು ಅಧಿಕಾರಿಗಳು ತಿಳಿಸಿದ್ದಾರೆ. .....

Table 5. Output tagged sequence

ವಾಷಿಂಗ್ಟನ್/NEL (ಪಿಟಿಐ):/NEO ಮುಂಬರುವ/NONE ಲೋಕಸಭಾ/NE ಚುನಾವಣೆ/NE
ಬಳಿಕ/NONE ನರೇಂದ್ರ/NEPB ಮೋದಿಯೊಂದಿಗೆ./NEPE ಅಮೆರಿಕ/NEL ರಾಜತಾಂತ್ರಿಕ/NONE
ಕೆಲಸ/NONE ನಿರ್ವಹಿಸಲು/NONE ಸಿದ್ಧವಿದ್ದು/NONE ಇಲ್ಲಿ/NONE ವೀಸಾ/NETE
ಪ್ರಶ್ನೆಯೇ/NONE ಇಲ್ಲ/NONE ಎ◌ಂದು/NONE ಅಮೆರಿಕ/NEL ಸ್ಪಷ್ಟಪಡಿಸಿದೆ./NONE
ಮೋದಿ/NEP ವಿಶ್ವದ/NE ಅತಿ/NONE ದೊಡ್ಡ/NONE ಪ್ರಜಾಪ್ರಭುತ್ವ/NEtE ರಾಷ್ಟ್ರದ/NE
ನಾಯಕನಾದರೆ/NONE ಅವರ/NONE ಜೋತೆ/NONE ನಾವು/NONE ಕೆಲಸ/NETE
ಮಾಡುತ್ತೇವೆ/NONE ಎ◌ಂದು/NONE ಒಬಾಮಾ/NEP ಆಡಳಿತದ/NONE ಹಿರಿಯ/NETE
ಅಧಿಕಾರಿಗಳು/NE ತಿಳಿಸಿದ್ದಾರೆ./NONE ಭಾರತೀಯ/NEOB ಜನತಾ/NEOI ಪಕ್ಷ/NEOE
ಮೋದಿ/NEP ಅವರನ್ನು/NONE ಪ್ರಧಾನಿ/NED ಅಭ್ಯರ್ಥಿ/NE ಎ◌ಂದು/NONE
ಘೋಷಿಸಿದೆ./NONE ಒಂದು/NEN ವೇಳೆ/NETI ಆ/NONE ಪಕ್ಷ/NE ಅಧಿಕಾರಕ್ಕೆ/NONE
ಬಂದರೆ/NONE ಮೋದಿಗೆ/NEP ಗೌರವ/NETE ನೀಡಬೇಕು./NONE ಹಾಗಾಗಿ/NONE
ಯಾವುದೇ/NONE ವೀಸಾ/NETE ಸಮಸ್ಯೆ/NONE ಉದ್ಭವಿಸುವುದಿಲ್ಲ/NONE ಎ◌ಂದು/NONE
ಅಧಿಕಾರಿಗಳು/NE ತಿಳಿಸಿದ್ದಾರೆ./NONE .....

As already mentioned, SVM model of Kannada NERC is trained using a tagged corpus of size 1, 51, 440 tokens. A sample input test sequence is given in Table 4. The SVM model generates the tagged output sequence as shown in Table 5. It can be noted that the Table 4 is a subset of actual test corpus of 7,000 tokens and the corresponding output tagged sequence is also the subset of the actual tagged sequence of 7,000 tokens. The performance of the designed SVM model is measured using various evaluation measurements as discussed in Section 5.

Table 6. Corpus size and program Run time

The training set size for the model	1,51,440 words
Total number of samples treated by the classifier	1,51,440 words
Total number of features extracted by the classifier	33273 (symbols)
Feature extraction Time (Training of SVM model)	1244.594 seconds
The test set size for the model	7000 words
Total number of features of test set	5775 (symbols)
Feature extraction Time for test data	2.031 seconds

Table 7. Results of 10 fold cross validation

FOLDS	Precision %	Recall %	F1 %	Support
1	81	81	81	15144
2	84	83	83.5	15144
3	85	83	84	15144
4	88	87	87.5	15144
5	85	84	84.5	15144
6	88	87	87.5	15144
7	79	77	78	15144
8	77	76	76.5	15144
9	83	82	82.5	15144
10	87	85	86	15144
Average/Total	83.7	82.5	83.1	151440

From Table 6 it can be noted that the execution time depends on the size of the input test corpus sequence. Table 7 shows the scores of individual folds and the combined scores of all the ten folds.

Table 8. Confusion matrix of the experiment

		PREDICTED CLASS																
		NEP	NEPBIE	NEL	NELBIE	NEO	NEOBIE	NED	NETE	NETP	NETO	NEB	NEM	NEN	NETI	NEA	NE	NONE
ACTUAL CLASS	NEP	189	4	2	0	0	0	1	0	0	0	0	0	0	0	0	0	128
	NEPBIE	3	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
	NEL	1	2	99	0	0	0	0	0	0	1	0	0	0	0	0	2	43
	NELBIE	0	1	1	6	0	0	0	0	0	0	0	0	0	0	0	0	4
	NEO	0	1	0	0	13	3	0	0	0	0	0	0	0	0	0	2	9
	NEOBIE	0	0	1	0	1	9	0	1	0	0	0	0	0	0	0	2	6
	NED	0	2	0	0	0	0	27	0	0	0	0	0	0	0	0	2	13
	NETE	1	0	0	0	0	0	0	24	0	0	0	1	0	3	0	3	71
	NETP	2	0	0	0	1	0	0	0	11	0	0	0	0	0	0	1	12
	NETO	0	1	0	0	2	1	0	5	0	34	0	0	0	0	0	2	54
	NEB	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	7
	NEM	1	0	0	0	0	0	0	0	0	0	0	24	3	1	0	0	39
	NEN	0	2	0	0	0	0	0	0	0	0	0	0	55	1	0	1	92
	NETI	0	1	1	0	0	0	0	0	0	0	0	0	3	38	0	0	12
	NEA	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5	0	5
	NE	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	247	147
	NONE	8	4	2	1	1	5	0	1	6	0	0	4	16	1	0	71	5349

From the confusion matrix, the calculation of P, TPR, and FPR are done as follows:

Consider named entity NEP as Positive class and all others entities as negative class:

TP for NEP class = 189 (first element of primary diagonal)

FP for NEP class = 18 (first column sum excluding TP value of 189)

FN for NEP class = 135 (first row sum excluding TP value of 189)

TN = 5950 (diagonal elements sum of confusion matrix excluding TP value of 189)

Precision (P) = TP / (TP + FP) = 189 / (189+18) = 0.91

Recall(R/TPR) = TP / (TP + FN) = 189 / (189+135) = 189/324 = 0.58

F1-Measure = 2PR / (P + R) = 2x 0.91x 0.58 / (0.91 + 0.58) = 1.0556/1.49 = 0.71

FPR = FP/ (FP+TN) = 18/ (18+5950) = 18/5968 = 0.0030

Similarly the results are calculated for all the NEs and tabulated in Table 10 and Table 11.

Table 9. Total number of NE's in training corpus

Named Entity (NE)	Tag	Tag label	Support
Person	NEP	0	37181
	NEPB	1	8222
	NEPI	2	4338
	NEPE	3	9216
Location	NEL	4	21419
	NELB	5	126
	NELI	6	78
	NELE	7	164
Organization	NEO	8	1090
	NEOB	9	254
	NEOI	10	280
	NEOE	11	202
Designation	NED	12	1495
Term	NETE	13	1065
Title-Person	NETP	14	1587
Title-Object	NETO	15	593
Brand	NEB	16	185
Measurement	NEM	17	1425
Number	NEN	18	946
Time	NETI	19	800
Abbreviation	NEA	20	5218
Noun entity	NE	21	32724
Not a NE	NONE	22	151440

Table 9 indicates the count of different named entitied in the whole training corpus created manually for this work.

Table 10. Classification Results of Test-set Corpus using linear kernel

Named Entity (NE)	Tag	Tag label	Precision	Recall	F1 - score	Support
Person	NEP	0	0.91	0.58	0.71	324
	NEP (BIE)	1 2 3	0.47	0.49	0.48	35
Location	NEL	4	0.93	0.67	0.78	148
	NEL (BIE)	5 6 7	0.86	0.50	0.63	12
Organization	NEO	8	0.72	0.46	0.56	28
	NEO (BIE)	9 10 11	0.50	0.45	0.47	20
Designation	NED	12	0.96	0.61	0.75	44
Term	NETE	13	0.77	0.23	0.35	103
Title-Person	NETP	14	0.65	0.41	0.50	27
Title-Object	NETO	15	0.97	0.34	0.50	99
Brand	NEB	16	1.00	0.22	0.36	9
Measurement	NEM	17	0.83	0.35	0.49	68
Number	NEN	18	0.71	0.36	0.48	151
Time	NETI	19	0.86	0.69	0.77	55
Abbreviation	NEA	20	1.00	0.45	0.62	11
Noun entity	NE	21	0.74	0.62	0.67	397
Not A NE	NONE	22	0.89	0.98	0.93	5469
Average /Total			87%	88%	87.5%	**7000**

It can be seen that the input test sequence is a good mix of all types of possible named entities. We have mixed single word named entities and multiword (beginning, internal and End, BOE) person names, location names and organization names in the corpus. So, it is inferred that the model is well tested for all kinds of possible classification opportunities. Table 10 and Table 11 give the performance of the model indicating the classification ability of the model and the error analysis respectively.

Table 11. Error analysis

Named Entity (NE)	Tag	Tagl abel	FPR	Support
Person	NEP	0	0.0030	324
	NEP (BIE)	1 2 3	0.0031	35
Location	NEL	4	0.0013	148
	NEL (BIE)	5 6 7	0.0002	12
Organization	NEO	8	0.0008	28
	NEO(BIE)	9 10 11	0.0015	20
Designation	NED	12	0.0002	44
Term	NETE	13	0.0011	103
Title-Person	NETP	14	0.0010	27
Title-Object	NETO	15	0.0002	99
Brand	NEB	16	0.0000	9
Measurement	NEM	17	0.0008	68
Number	NEN	18	0.0036	151
Time	NETI	19	0.0010	55
Abbreviation	NEA	20	0.0000	11
Noun entity	NE	21	0.0144	397
Not a NE	NONE	22	0.4509	5469
Average /Total			2.84%	7000

It is interesting that the proposed model works with higher F1-measure 87.5% on a test corpus of 7000 tokens with linear kernel SVM. The time taken for extraction of the features by SVM training model is 1244.594 seconds for a training corpus size of 1, 51,440 tokens. Moreover, it can be seen that the testing time is very less of the order of 2.031 seconds, which mainly depends on the size of test corpus (7,000 words in this case). 10 fold cross validation results of the system in terms of Precision, Recall and F1-measure are 83.7%, 82.5% and 83.1% respectively.

The results of SVM model with different kernels is tabulated in Table 12 for different sizes of Test-Set. It is seen that the SVM model with linear kernel gives highest F1-Score of 87.5% on a test corpus of 7000 tokens

Table 12. F1-scores of SVM model with different kernels

Test-Set size in words	**F1-Score in %**			
	linear kernel	**Poly kernel**	**Rbf kernel**	**Sigmoid kernel**
7,000	**87.5**	**68.5**	**51**	**51**
11,000	**82**	**51**	**59.7**	**59.7**
15,000	**81**	**51**	**51**	**51**
20,000	**74.5**	**34.4**	**44.6**	**34.4**
30,000	**71.9**	**20.5**	**40.9**	**20.5**
40,000	**70.9**	**12.5**	**32.9**	**12.5**
50,000	**68.4**	**6.6**	**32.5**	**6.6**

7 Conclusion

Natural Language Processing is an important research area containing challenging issues to be investigated. NERC is a class of NLP which is used for extracting named entities from unstructured data. In this context, this paper focuses on NERC in Kannada language, as it is found that little work is done in this area. In this direction, we have conducted an extensive survey in the related area of NLP and based on the survey, we have proposed a problem and the methodology that has been formulated. Various modeling techniques are investigated, out of which design of supervised SVM is reported here. We have developed an efficient model which is trained on a corpus consisting of 1, 51,440 words. From the test corpus, variety of test samples are chosen randomly and fed as input to the SVM model with different kernels. It is interesting to note that the model recognizes the named entities with an average F1-measure of 87.5% and 10 fold cross validation F1-measure of 83.1% for a test corpus of 7000 tokens with linear kernel. The false positive rate of the algorithm is 2.84% for a test corpus of 7000 tokens with linear kernel.

REFERENCES

[1]. Elizabeth D Liddy, 2001. Natural Language Processing. In Encyclopedia of Library and Information Science.2nd edition.

[2]. James Allen, 2007. Natural Language Understanding. Pearson Publication Inc., 2nd edition.

[3]. Kavi Narayana Murthy, 2006. Natural Language Processing. Ess Ess Publications for Sarada Ranganathan Endowment for Library Science, Bangalore, India, 1st edition.

[4]. Gobinda G. Chowdhury, 2003. Natural Language Processing, Annual Review of Information Science and Technology. 37(1):51-89.

[5]. Kashif Riaz, 2010. Rule-based Named Entity Recognition in Urdu. In Proceedings of the 2010 Named Entities Workshop, pages 126-135. Association for Computational Linguistics.

[6]. Khaled Shaalan and Hafsa Raza, 2007. Person Name Entity Recognition for Arabic. In roceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, pages 17-24. Association for Computational Linguistics.

[7]. Yassine Benajiba, Mona T Diab and Paolo Rosso, 2009. Using Language Independent and Language Specific Features to Enhance Arabic Named Entity Recognition. Int. Arab J. Inf. Technol., 6(5):463-471.

[8]. Padmaja Sharma, Utpal Sharma and Jugal Kalita, 2010. The First Steps towards Assamese Named Entity Recognition. Brisbane Convention Center, 1:1-11.

[9]. Asif Ekbal and Sivaji Bandyopadhyay, 2009. A Conditional Random Field Approach for Named Entity Recognition in Bengali and Hindi. Linguistic Issues in Language Technology, 2(1).

[10]. Asif Ekbal and Sivaji Bandyopadhyay, 2009. Named Entity Recognition in Bengali. A Multi-engine Approach. Northern European Journal of Language Technology, 1(2):26-58.

[11]. Asif Ekbal and Sivaji Bandyopadhyay, 2008. Bengali named entity recognition using support vector machine. In IJCNLP, pages 51-58.

[12]. Maksim Tkachenko, Andrey Simanovsky and St Petersburg, 2012. Named entity recognition: Exploring features. In Proceedings of KONVENS, pages 118-127

[13]. [Ashwini A Shende and Avinash J Agrawa, 2012. Domain specific named entity recognition. Proceedings of the International Conference on Advances in Computer, Electronics and Electrical Engineering, ISBN: 978-981-07-1847-31, doi:10.3850/978-981- 07-1847-3 P0999:484-487.

[14]. David Nadeau, Peter Turney and Stan Matwin, 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. Published at the 19th Canadian Conference on Artificial Intelligence.

[15]. Sujan Kumar Saha, Partha Sarathi Ghosh, Sudeshna Sarkar and Pabitra Mitra, 2008 . Named entity recognition in Hindi using maximum entropy and transliteration. Research journal on Computer Science and Computer Engineering with Applications, pages 33-41.

[16]. Deepti Chopra, Nusrat Jahan and Sudha Morwal, 2012. Hindi named entity recognition by aggregating rule based heuristics and hidden markov model. International Journal of Information, 2(6).

[17]. Sujan Kumar Saha, Shashi Narayan, Sudeshna Sarkar and Pabitra Mitra, 2010. A composite kernel for named entity recognition. Pattern Recognition Letters, 31(12):1591-1597, doi:10.1016/ j.patrec.2010.05.004.

[18]. Sudha Morwal and Nusrat Jahan, 2013. Named entity recognition using hidden markov model (hmm): An experimental result on Hindi, Urdu and Marathi languages. International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), 3(4):671-675.

[19]. Erik F Tjong Kim Sang and Fien De Meulder, 2003. Introduction to the conll-2003 shared task: Languageindependent named entity recognition. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, pages 142-147. Association for Computational Linguistics.

[20]. Asif Ekbal and Sivaji Bandyopadhyay, 2010. Named entity recognition using support vector machine: A language independent approach. International Journal of Electrical, Computer, and Systems Engineering, 4(2):155-170.

[21]. Kishorjit Nongmeikapam, Tontang Shangkhunem, Ngariyanbam Mayekleima Chanu, Laishram Newton Singh, Bishworjit Salam and Sivaji Bandyopadhyay, 2011. CRF based name entity recognition (ner) in Manipuri: A highly agglutinative Indian language. In Emerging Trends and Applications in Computer Science (NCETACS), 2nd National Conference on, pages 1-6. IEEE.

[22]. Thoudam Doren Singh, Kishorjit Nongmeikapam, Asif Ekbal and Sivaji Bandyopadhyay, 2009. Named entity recognition for Manipuri using support vector machine. In PACLIC, pages 811-818.

[23]. S Pandian, Krishnan Aravind Pavithra and T Geetha, 2008. Hybrid three-stage named entity recognizer for Tamil. INFOS.

[24]. R Vijayakrishna and Sobha Lalitha Devi, 2008. Domain focused named entity recognizer for Tamil using conditional random fields. In IJCNLP, pages 59-66.

[25]. G.V.S.Raju B.Srinivasu, Dr.S.Viswanadha Raju and K.S.M.V.Kumar, 2010. Named entity recognition for Telugu using maximum entropy model. Journal of Theoretical and Applied Information Technology (JATIT), 13:125-130.

[26]. Dr. A. Vinaya Babu, Dr. A. Govardhan, B. Sasidhar and P. M. Yohan, 2011. Named entity recognition in Telugu language using language dependent features and rule based approach. International Journal of Computer Applications (0975-888), 22(8):30-34.

[27]. [S Amarappa, Dr. S V Sathyanarayana, 2013. "A Hybrid approach for Named Entity Recognition, Classification and Extraction (NERCE) in Kannada Documents". Proceedings of the International Conference on Multimedia Processing, Communication and Information Technology (MPCIT-2013). Book Series: Advances in Engineering and Technology Series, IDES publications, DOI: 03.AETS.2013.4.91, ISBN: 2214 - 0344, Volume: 4, Page(s):173-179.

[28]. S Amarappa and S V Sathyanarayana, 2013. Named entity recognition and classification in Kannada language. International Journal of Electronics and Computer Science Engineering, 2(1):281–289.

[29]. S Amarappa, and S V Sathyanarayana, 2015. Kannada Named Entity Recognition and Classification (NERC) based on Multinomial Naïve Bayes (MNB) Classifier. International Journal on Natural Language Computing (IJNLC), DOI: 10.5121/ijnlc.2015.4404, Vol. 4, No.4, Pages 39-52.

[30]. S Amarappa, and. S V Sathyanarayana, 2015. Kannada Named Entity Recognition and classification (NERC) based on Conditional Random fields (CRF). Second International Conference on Emerging Research on Electronics, Computer Science and Technology (ICERECT-2015), PES College of Engineering, Mandya, India. 978-1-4673-9563-2/15/$31.00 ©2015 IEEE.

[31]. David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification, National Research Council, Canada / New York University.