DataScience
Language Identification Task - Complete Guide

This file will provide complete guidance for the completion of the given task and to replicate the results obtained by Bhattu sir in his research work.

To read paper by Bhattu sir, click here

I have attached the snapshots for clear understanding.

WHAT TO DO????????

Well… The task is **identifying the languages in a given sentence , word wise .**

Sample Input-Output :

1) Input Sentence : " Apna time ayega "
   Output : Hindi, English, Hindi
2) Input Sentence : " Life is very short nanba "
   Output : English, English, English, English, Tamil

We will show these results in terms of Accuracy / F1-Scores.

WHY TO DO????????

In present social media like WhatsApp, Instagram,Twitter everyone will use mixed languages in chatting, comments ..etc and like in E-Commerce websites, while giving reviews.

For product improvement, Business Analysts at companies from Amazon, FlipKart will get insights from those comments using **TextProcessing.** And for that, this Language Identification is the preliminary task to understand the text.

WHY MACHINE LEARNING FOR THIS TASK?????

We have plenty of languages and vocabulary. It's highly impossible to store all the language dictionaries and hence we are training a model with some languages so that the model will get insights from training data, And we are using that model for prediction of languages in any given sentence..

HOW TO DO?????????

This task is divided into two major stages

1) Sentence level Classification
2) Word level Classification

We have 8 Indian Languages(bengali, gujarathi,hindi,kannada,malayalam,marati,tamil,telugu) and English, totally - 9

For this task, we have some constraints

1. Any sentence can have **at most** mixing of **two** languages
2. If no. of languages > 1, English is compulsory language

   I.e. , In a given sentence,

   English , Hindi combination - possible

   Telugu , Hindi - not possible

   Telugu, English, Tamil - not possible

STAGE - 1 : Sentence Level Classification

The possible language mixing combinations are

1. en_be_
2. en_gu_
3. en_hi_
4. en_kn_
5. en_ml_
6. en_mr_
7. en_te_
8. en_ta_
9. en_ (Only te_ or only hi_ … can be addressed in word level)

Hence the given problem is formulated as a 9-class classification problem , which predicts Sentence level tags.

At this stage,

1) Input - “Apna time ayega”
   Output - hi_en_
2) Input - “Life is very short nanba”
   Output - ta_en_

IMPLEMENTATION:

For this, we have to train NaiveBayes, Logistic Regression models with data.
Actual data : InputTraining.txt, InputTesting.txt
And for this task, n-grams will help because:

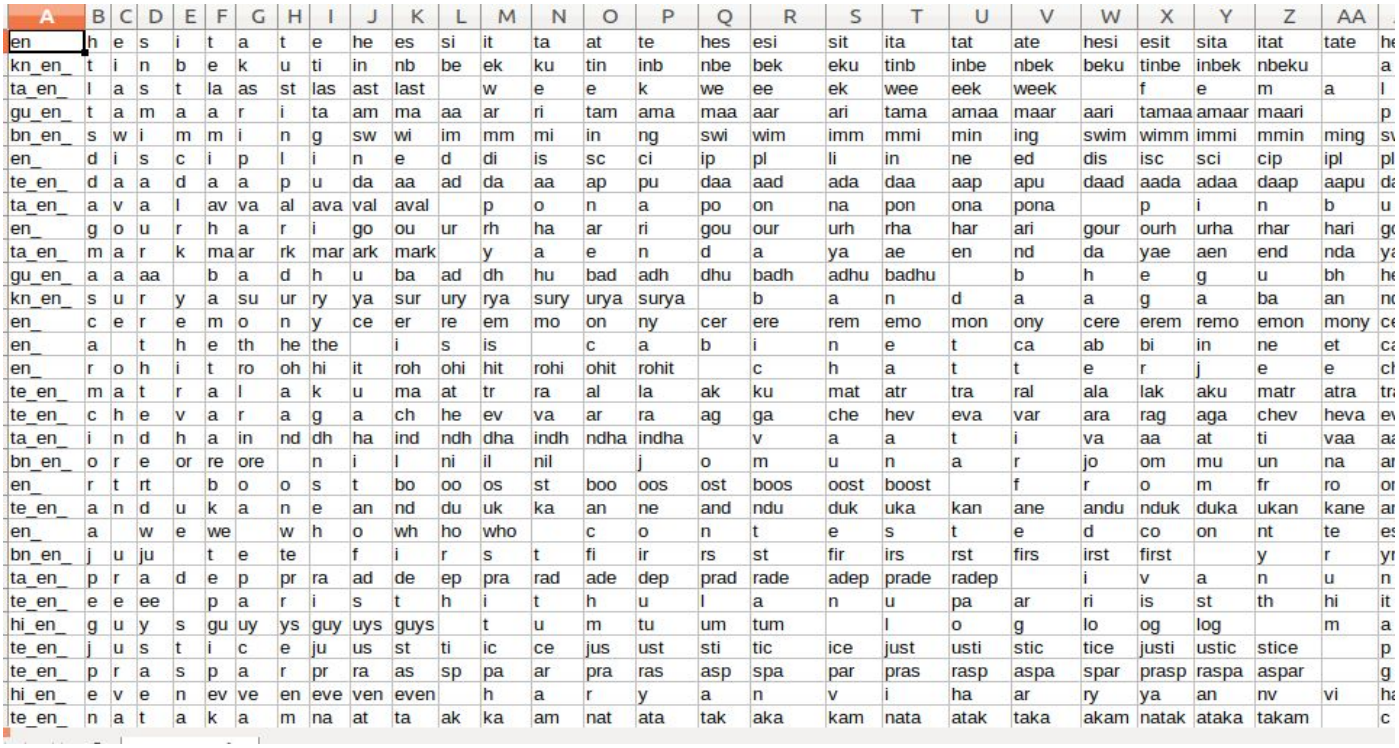
Consider if training has the word “ippudu” and in testing we have the word “eppudu”, they both will belong to the same language as per that language rules. Hence, we need to consider **n-grams of text, throughout this task.**

Sir, has already formed n-grams for some of the data and we can directly use that.

Training data : ngrams_train.txt
Testing data : ngrams_test.txt

-- (available in sir's github or all the files are available in the folder attached at the end of this document)

Data in ngrams_train.txt:



en_	h	e	s	i	t	a	t	e	he	es	si	it	ta	at	te	hes	esi	sit	ita	tat	ate	hesi	esit	sita	itat	tate	he
kn_en_	t	i	n	b	e	k	u	ti	in	nb	be	ek	ku	tin	inb	nbe	bek	eku	tinb	inbe	nbek	beku	tinbe	inbek	nbeku		a
ta_en_	l	a	s	t	la	as	st	las	ast	last		w	e	e	k	we	ee	ek	wee	eek	week		f	e	m	a	l
gu_en_	t	a	m	a	a	r	i	ta	am	ma	aa	ar	ri	tam	ama	maa	aar	ari	tama	amaa	maar	aari	tamaa	amaar	maari		p
bn_en_	s	w	i	m	m	i	n	g	sw	wi	im	mm	mi	in	ng	swi	wim	imm	mmi	min	ing	swim	wimm	immi	mmin	ming	sv
en_	d	i	s	c	i	p	l	i	n	e	d	di	is	sc	ci	ip	pl	li	in	ne	ed	dis	isc	sci	cip	ipl	pl
te_en_	d	a	a	d	a	a	p	u	da	aa	ad	da	aa	ap	pu	daa	aad	ada	daa	aap	apu	daad	aada	adaa	daap	aapu	da
ta_en_	a	v	a	l	av	va	al	ava	val	aval		p	o	n	a	po	on	na	pon	ona	pona		p	i	n	b	u
en_	g	o	u	r	h	a	r	i	go	ou	ur	rh	ha	ar	ri	gou	our	urh	rha	har	ari	gour	ourh	urha	rhar	hari	g
ta_en_	m	a	r	k	ma	ar	rk	mar	ark	mark		y	a	e	n	d	a	ya	ae	en	nd	da	yae	aen	end	nda	ya
gu_en_	a	a	aa		b	a	d	h	u	ba	ad	dh	hu	bad	adh	dhu	badh	adhu	badhu		b	h	e	g	u	bh	he
kn_en_	s	u	r	y	a	su	ur	ry	ya	sur	ury	rya	sury	urya	surya		b	a	n	d	a	a	g	a	ba	an	n
en_	c	e	r	e	m	o	n	y	ce	er	re	em	mo	on	ny	cer	ere	rem	emo	mon	ony	cere	erem	remo	emon	mony	ce
en_	a		t	h	e	th	he	the		i	s	is		c	a	b	i	n	e	t	ca	ab	bi	in	ne	et	ca
en_	r	o	h	i	t	ro	oh	hi	it	roh	ohi	hit	rohi	ohit	rohit		c	h	a	t	t	e	r	j	e	e	ch
te_en_	m	a	t	r	a	l	a	k	u	ma	at	tr	ra	al	la	ak	ku	mat	atr	tra	ral	ala	lak	aku	matr	atra	tra
te_en_	c	h	e	v	a	r	a	g	a	ch	he	ev	va	ar	ra	ag	ga	che	hev	eva	var	ara	rag	aga	chev	heva	ev
ta_en_	i	n	d	h	a	in	nd	dh	ha	ind	ndh	dha	indh	ndha	indha		v	a	a	t	i	va	aa	at	ti	vaa	aa
bn_en_	o	r	e	or	re	ore		n	i	l	ni	il	nil		j	o	m	u	n	a	r	jo	om	mu	un	na	ar
en_	r	t	rt		b	o	o	s	t	bo	oo	os	st	boo	oos	ost	boos	oost	boost		f	r	o	m	fr	ro	or
te_en_	a	n	d	u	k	a	n	e	an	nd	du	uk	ka	an	ne	and	ndu	duk	uka	kan	ane	andu	nduk	duka	ukan	kane	ar
en_	a		w	e	we		w	h	o	wh	ho	who		c	o	n	t	e	s	t	e	d	co	on	nt	te	es
bn_en_	j	u	ju		t	e	te		f	i	r	s	t	fi	ir	rs	st	fir	irs	rst	firs	irst	first		y	r	yr
ta_en_	p	r	a	d	e	p	pr	ra	ad	de	ep	pra	rad	ade	dep	prad	rade	adep	prade	radep		i	v	a	n	u	n
te_en_	e	e	ee		p	a	r	i	s	t	h	i	t	h	u	l	a	n	u	pa	ar	ri	is	st	th	hi	it
hi_en_	g	u	y	s	gu	uy	ys	guy	uys	guys		t	u	m	tu	um	tum		l	o	g	lo	og	log		m	a
te_en_	j	u	s	t	i	c	e	ju	us	st	ti	ic	ce	jus	ust	sti	tic	ice	just	usti	stic	tice	justi	ustic	stice		p
te_en_	p	r	a	s	p	a	r	pr	ra	as	sp	pa	ar	pra	ras	asp	spa	par	pras	rasp	aspa	spar	prasp	raspa	aspar		g
hi_en_	e	v	e	n	ev	ve	en	eve	ven	even		h	a	r	y	a	n	v	i	ha	ar	ry	ya	an	nv	vi	ha
te_en_	n	a	t	a	k	a	m	na	at	ta	ak	ka	am	nat	ata	tak	aka	kam	nata	atak	taka	akam	natak	ataka	takam		c

We need to pre-process this data to fit into the model.
Pre-processing:

1) Divide data into target and feature variables
2)Encode target variable i.e., 9 class variable from 0-8

```
df_train = pd.read_csv('ngramsTrain.txt',header=None)
df_test = pd.read_csv('ngramsTest.txt',header=None)

y_train = df_train[0]
x_train = df_train[1]
y_test = df_test[0]
x_test = df_test[1]

#Encoding 9 classes for classification
mapping =
{"bn_en_":0,"en_":1,"gu_en_":2,"hi_en_":3,"kn_en_":4,"ml_en_":5,"mr_en_":6,"ta_en_":7,"te_en_":8}
df_train = df_train.replace(mapping)
df_test = df_test.replace(mapping)
```

3) Feature variable has text. In Order to feed that data to model, it should be in numeric format. And there is a module called **CountVectorizer** to perform that task.
Refer to a small video on CountVectorizer on YouTube.

For that, we need to import that from sklearn and fit the data and can transform it.

```
cv = CountVectorizer()
cv.fit(x_train)
x_train = cv.transform(x_train)
x_test = cv.transform(x_test)
```

At this stage, data is preprocessed and ready to train the models.

Import the classification models from sklearn and train them and measure their accuracy.

```
nb = MultinomialNB()
nb.fit(x_train,y_train)
y_pred = nb.predict(x_test)          #y_pred contains predicted labels and y_test contains actual labels
print("F1 Score of Naive bayes for sentence classifier is ",metrics.accuracy_score(y_test,y_pred))
```

Apply in the same way for logistic regression model too..
(Logistic regression is also known as Maximum Entropy model)

This is the end of Stage-1.
After executing, u may end up with accuracies like 71%.
attachment: /home/vsriharshini/Pictures

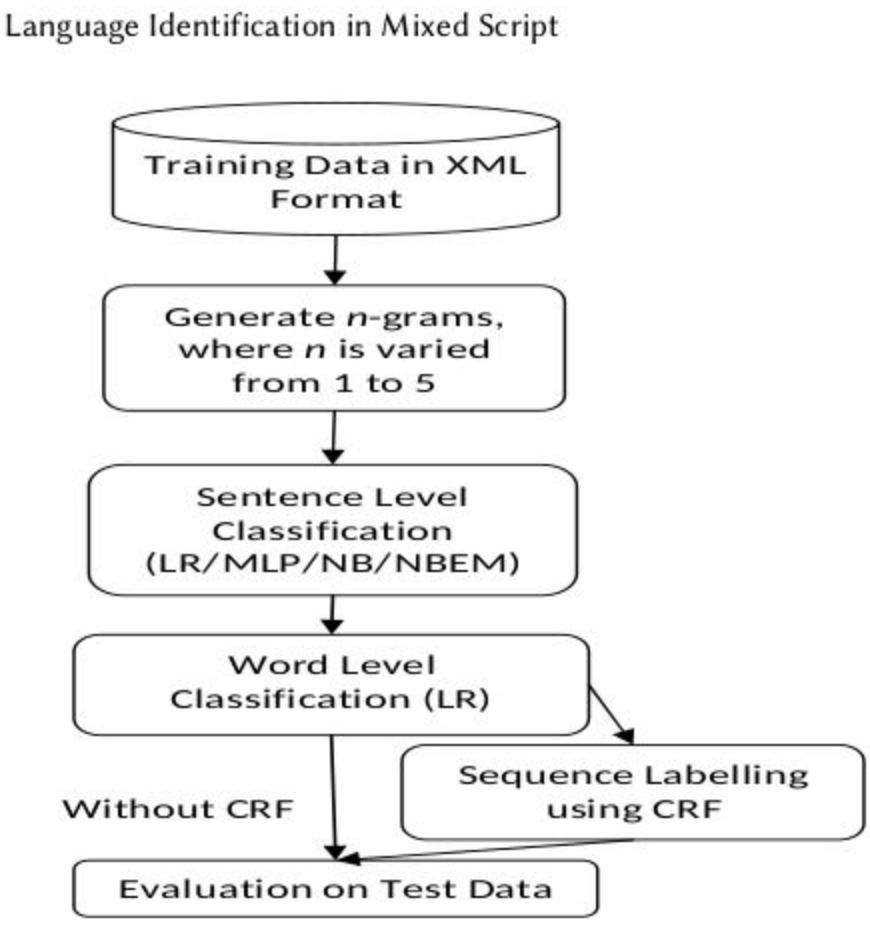


Figure 1: Overview of the approach followed

STAGE - 2 : Word Level Classification

Till now, given any sentence, sentence level classifier predicts the sentence level tag.
Now, it's time to predict word level tags.

For a given sentence “Apna time ayega”, stage1 predicts the tag as hi_en_. Now we have to say
whether first word “Apna” in hindi/english
whether second word “time” in hindi/english
whether third word “ayega” in hindi/english.

For this, the suggested approach is to build 8 binary classifiers for determining whether a given word is in the language ln/en when the tag predicted by upper stage is ln_en_.

Hence, now the task is to build **binary classifiers for 8 Indian languages Vs English.**

IMPLEMENTATION:

For building these models,
Training and testing data : Vocabulary files of 9 languages (For files, refer github of Bhattu sir or the folder attached by me)

Pre-processing : Data preprocessing is the same as for the above stage.
For eg, while building en_te_ classifier,

1. Merge eng.txt and telugu.txt
2. Label for all words from eng.txt -> 0
   Label for all words from telugu.txt -> 1
3. Convert the n-grams into numeric using CountVectorizer.
4. Split using train-test split from sklearn
5. Fit the train data into model and predict on test data.

But here, n-grams are not available except for the english.txt . Hence, we need to convert them into n-grams before converting them into numeric features using CountVectorizer.

```
def ngram_generator(n,word):
        i=0
        n_grams=''
        j=1
        while(j<=n):
        i=0
        while(i<=len(word)-j):
        n_grams+=word[i:i+j]+' '
        i+=1
        j+=1
        return n_grams
```

By following the same stage1 procedure, build 8 binary classifiers by train and test split of vocabulary files.
This will result in F1-scores >85% for all the 8 classifiers.

Hence , after this step, given sentence “ Apna time ayega “ and tag hi_en_
Stage 2 invokes Hindi-English binary classifier to predict word level tags and hence predicts tags for words as “Apna” - Hindi
“Time” - English
“Ayega” - Hindi
Hence the task is done.

To predict accuracies, we need to merge the execution of Stage-1 and Stage-2 on testing data after building the respective classifiers.

MERGING STAGE1 AND STAGE2 on TESTING DATA:

Testing data and the word level tags are available in files

1. InputTesting.txt
2. AnnotationsTest.txt

but in xml format.

InputTesting.txt

```
<data>
        <utterance id="1">
                and lightening exabits in videos http://t.co/s22zrne4bx gigabits transfer ll delhi
watch air will of be #aapstorm data #aapsweep
        </utterance>
        <utterance id="2">
                12. rogi : doctor , ondu vishayadhalli neevu tumbhaa lucky .
        </utterance>
        <utterance id="3">
                rt trending trendsmapindia is india #aapstorm http://t.co/gpohawcwwo in now
        </utterance>
        <utterance id="4">
                mahesh babu . davudu lanti vadu .......
        </utterance>
        <utterance id="5">
                ninna malligeya ka0pige
        </utterance>
        <utterance id="6">
                Amar ubuntute sudo apt gete kono download hochhena keno
        </utterance>
        <utterance id="7">
                Aman Sharma ( Unknown )
        </utterance>
```

AnnotationsTesting.txt

```
<data>
        <utterance id="1">
                en en en en en X en en en NE en en en en en X en X
        </utterance>
        <utterance id="2">
                X kn X en X kn kn kn kn en X
        </utterance>
        <utterance id="3">
                X en en en NE X X en en
        </utterance>
        <utterance id="4">
                NE_P NE_P X te te te X
        </utterance>
        <utterance id="5">
                kn kn X
        </utterance>
```

In Order to extract data from those format we need to scrap using **beautifulsoup or XML scraper**

```
import xml.etree.cElementTree as etree

xmlDoc = open('AnnotationTesting.xml', 'r')
xmlDocData = xmlDoc.read()
xmlDocTree = etree.XML(xmlDocData)
tags_list = []
for ingredient in xmlDocTree.iter('data'):
        for i in range(len(ingredient)):
        tags_list.append(ingredient[i].text)
```

Similarly extract sentences and tags from those files and iteratively perform above two demonstrated stages on each incurred sentence.

NOTE : For best results , remove tags like NE_P, X and their respective words.

And finally, find F1-Scores and classification reports.

That will give results approximate to this…..

Table 4: Strict F-measure of classifiers v

Language	MaxEnt	NaiveBayes
English	0.83	0.8343
Bengali	0.7988	0.8263
Gujarati	0	0
Hindi	0.6426	0.7063
Kannada	0.8171	0.7267
Malayalam	**0.8345**	0.6667
Marati	0.7486	0.7848
Tamil	0.8362	0.8291
Telugu	0.6131	0.6248

(Results from research paper)

And in order to improve these accuracies, the concept of CRF came into picture.

The whole process depicted in this pdf is **Week1 task**