Word Prediction for Text Typing in Telugu

**1. Introduction:**

This project is Data Science project

The goal of the Data Science project is to use Natural Language Processing NLP theory , to develop an application that predict the next word for a device user to type in (Next-Word-Prediction).

NLP : Up to 1980, most NLP systems were based on complex sets of hand written rules, starting the late 1980s, due to the increase of computational power, there was a revolution in NLP with the introduction of machine learning algorithms for language processing Part of speech tagging introduced the use of hidden Markov Models to NLP, since , research has focused on statistical models which make soft probabilistic decisions based on attaching real-valued weight to the features making the input data.

**2. Abstract:**

A dominant technology in Natural Language Processing is called n-grams models and they work well for word prediction.

NLP models, and all models, are as reliable as the data they are trained on is, the n-grams (Which are the model features) will be learned from the training data.

In this report I will present the following topics:

Summarize the statistics of the project text data sets.

Describe some data characteristics.

Discuss my plan for developing a prediction algorithm and a shiny app.

**3. Synopsis:**

The task of the Capstone project is to build a model to predict the next word based on the previous words entered by the device user. The data is a collection of text documents coming from online collections of text data from Sakshi(Telugu text) and Telugu blogs.

The corpus (which is the set of documents) for my project will be limited to all utf-8 text files but in this I am doing with only Telugu text files only.

**4.N-grams models of languages:**

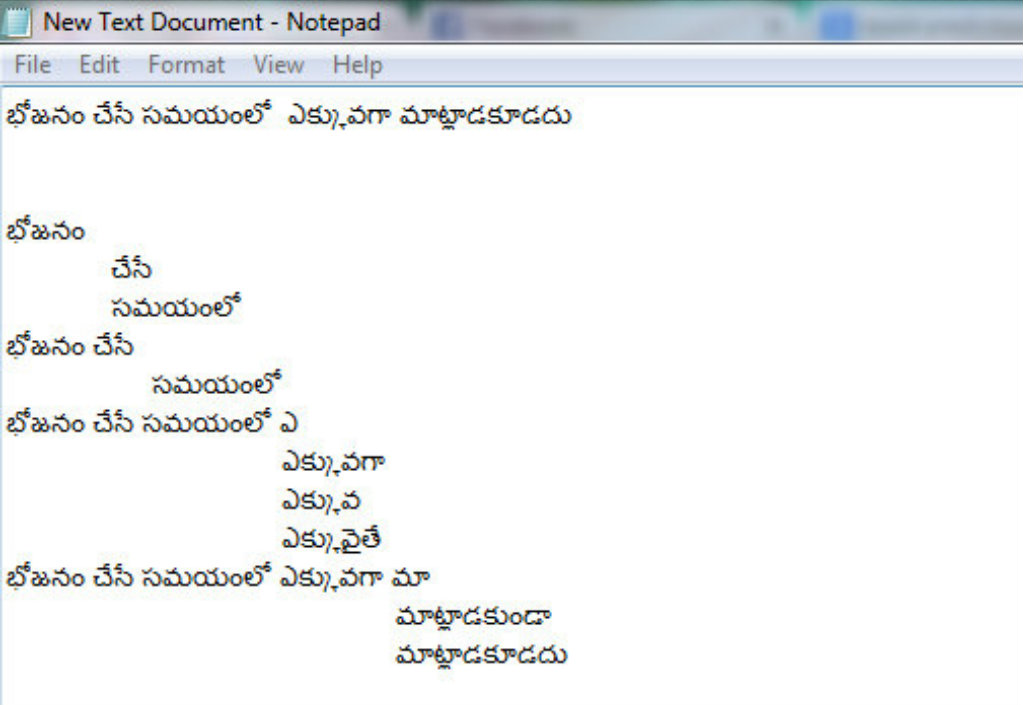
* Use the previous N-1 words in a sequence to predict the next word
* Language Model (LM)

Unigrams, bigrams, trigrams………..

* How do we train these models?Very large corpora

**5.Next word prediction:**

The task of the project is to build a model to predict the next word based on the previous words entered by the device user



**6. Packages:**

* **tm :**

This vignette gives a short introduction to text mining in R utilizing the text mining

Framework provided by the tm package. We present methods for data import, corpus

handling, preprocessing, metadata management,and creation of term-document

matrices.

* **Rweka :**

An R interface to Weka (Version 3.7.12).

Weka is a collection of machine learning algorithms for data mining

tasks written in Java, containing tools for data pre-processing,

classification, regression, clustering, association rules, and

visualization. Package RWeka contains the interface code, the Weka

jar is in a separate package RWekajars

* **Stringi:**

stringi allows for fast, correct, consistent, portable, and convenient character string/text processing in every locale and any native encoding. Owing to the use of the ICU library, the package provides R users with platform-independent functions known to Java, Perl, Python, PHP, and Ruby programmers.

* **Knitr :**

A General-Purpose Package for Dynamic Report Generation in R.

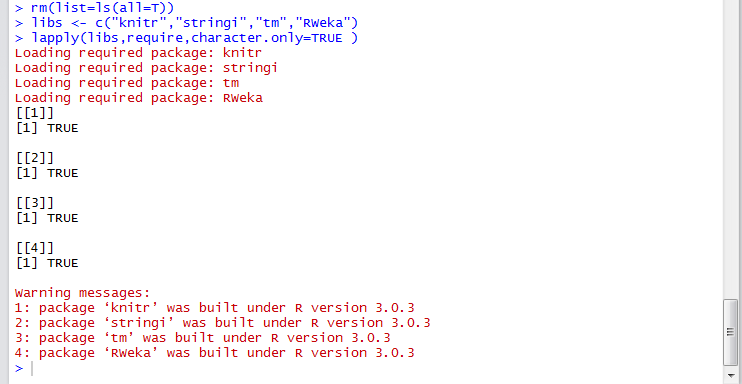
* **SnowballC :**

Snowball stemmers based on the C libstemmer UTF-8 library.

* **Wordcloud :**A word cloud (or tag cloud) can be an handy tool when you need to highlight the most commonly cited words in a text using a quick visualization. Of course, you can use one of the several on-line services, such as wordle or tagxedo , very feature rich and with a nice GUI. Being an R enthusiast, I always wanted to produce this kind of images within R and now, thanks to the recently released Ian Fellows' wordcloud package, finally I can!

In order to test the package I retrieved the titles of the XKCD web comics included in my RXKCD package and produced a word cloud based on the titles' word frequencies calculated using the powerful tm package for text mining (I know, it is like killing a fly with a bazooka!).

**7. Packages loading**: After loading the packages it looks like below:



**8.Statistical Estimators:**

Example:

Corpus: data from sakshi e-papers (Telugu text).

* Lines count:
* freqs2Lines <- length(freqs2)
* [1] 980 (from training data).
* Characters count:
* freqs2Nchar <- sum(nchar(freqs2,type="chars"))
* [1] 168949
* Words count:
* sum(stri\_count\_words(freqs2))
* [1] 21734
* Summary statistics :
* stri\_stats\_general(freqs2)

|  |  |  |  |
| --- | --- | --- | --- |
| Lines | LinesNEmpty | Chars | CharsNWhite |
| 980 | 980 | 168948 | 147950 |

* Given the observed training data …..

**9. Data cleaning:**

* Text cleaning Prior to corpus Tokenization and for an adequate text mining data need to be cleaned in order to extract significant features.
* Required package : tm
* tm\_map function:The cleaning function cleans as follow:
* remove no alphanumeric
* remove stop word
* remove URLs
* remove punctuation
* remove no English words
* remove numbers
* Strip white spaces Not all of them are necessary I will get back to this in next weeks.

doc.vec<- VectorSource(freqs2)

mycorpus<- Corpus(doc.vec)

summary(mycorpus)

mycorpus<- tm\_map(mycorpus, removePunctuation)

mycorpus<- tm\_map(mycorpus, removeNumbers)

mycorpus<- tm\_map(mycorpus, content\_transformer(tolower))

mycorpus<- tm\_map(mycorpus, stripWhitespace)

mycorpus<- Corpus(VectorSource(mycorpus ))

corp<- tm\_map(corp,stemDocument)

#preview the sample after cleaning

head(mycorpus)

tail(mycorpus)

**10. Probabilistic Language Modeling:**

* Manning &schutze. Statistical Natural Language Processing. 1999
* A sentence W is defined as a sequence of words w1, …, wn
* W = (w1,w2,w3,...,wn)
* The big (modeling) question is what is p(W)?
* Well, we know (chain rule):
  + p(W) = p(w1,w2,w3,...,wn) = p(w1)p(w2|w1)p(w3|w1,w2).............  
     p(wn|w1,w2,...,wn-1)
* Probability of next word wn in a sentence is: P(wn |w1, …, wn-1)
* a conditional probability
* The probability of the whole sentence is: P(W) = P(w1, …, wn)
* the chain rule of conditional probability
* These counts & probabilities form the language model
  + [for a given document collection (=corpus)].
* The model variables are discrete (counts).
* Only needs to deal with probability mass (not density).

**11. N-gram Language Models:**

* (n-1)th order Markov approximation ® n-gram LM:

**p(W) = Pi=1..n p(wi|wi-n+1,wi-n+2,...,wi-1)**

* In particular (assume vocabulary |V| = ):
* 0-gram LM: uniform model p(w) = 1/|V|
* 1-gram LM: unigram model p(w)
* 2-gram LM: bigram model p(wi|wi-1)
* 3-gram LM: trigram mode p(wi|wi-2,wi-1)
* 4-gram LM: tetragram model p(wi| wi-3,wi-2,wi-1)

**12. N-gram functions:**

* 3 ngrams tokenization functions for unigrams,bigrams and trigrams Set the default number of threds to use, it is needed for ngram function.

options(mc.cores=1)

uniTokenizer <- function(t) NGramTokenizer(t,Weka\_control(min=1,max=1))

biTokenizer <- function(t) NGramTokenizer(t,Weka\_control(min=2,max=2))

triTokenizer <- function(t) NGramTokenizer(t,Weka\_control(min=3,max=3))

**13.words,tokens and N-grams:**For example

Text with words: ఆరోజుచాలామందికిజాతీయగీతాన్నిగురించితెలియదు.

* Tokens (Token N-gram): ఆ,రోజు,చాలా,మందికి ,జాతీయ,గీతాన్ని,గురించి,తెలియదు
* Bigrams: ఆ రోజు,రోజు చాలా, చాలా మంది,మందికి జాతీయ,జాతీయ గీతాన్ని,

గీతాన్నిగురించి,గురించితెలియదు

* Trigrams: ఆ రోజు చాలా,రోజు చాలా మందికి, మందికి జాతీయ గీతాన్ని,

జాతీయ గీతాన్ని గురించి, గీతాన్ని గురించి తెలియదు

**14.Document TERM Matrix:**

Create Document Term Matrix and calculate frequencies using package tm and plot the 20 most frequent ngrams.

dtm <- DocumentTermMatrix(mycorpus, control=list(tokenize=uniTokenizer))

unifreq <- colSums(as.matrix(dtm)) #finding frequency of the dtm

barplot(tail(sort(unifreq),20),las=2,main="Top20unigrams",cex.main=1,cex.axis =0.75,horiz=TRUE)

Similarly we’ll find bigrams, trigrams, quadrigrams based on above functions.

**15.Frequency of words**:

The frequencies can be finding following functions

unifreq <- colSums(as.matrix(dtm)) #unigram frequencies

bifreq <- colSums(as.matrix(dtm)) #unigram frequencies

trifreq <- colSums(as.matrix(dtm)) #bigram frequencies

quadfreq <- colSums(as.matrix(dtm)) #quadrigram frequencies

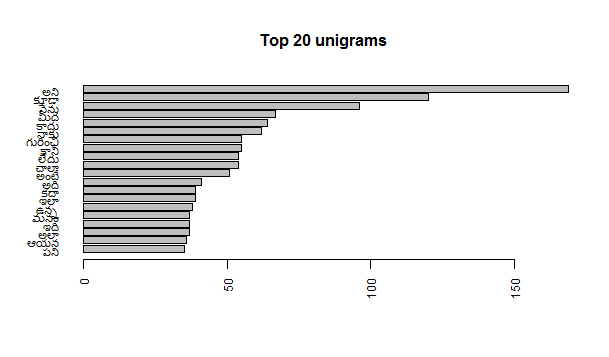
**16. plots** :

The bar plots can be work on following function :

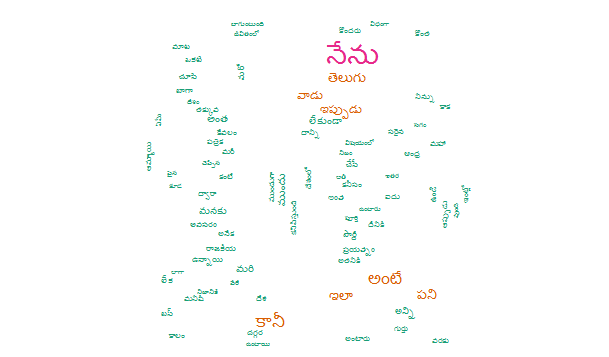
barplot(tail(sort(unifreq),20),las=2,main="Top 20

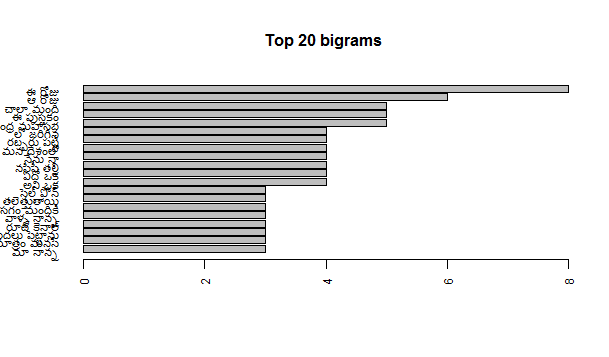
unigrams",cex.main=1,cex.axis =0.75,horiz=TRUE)

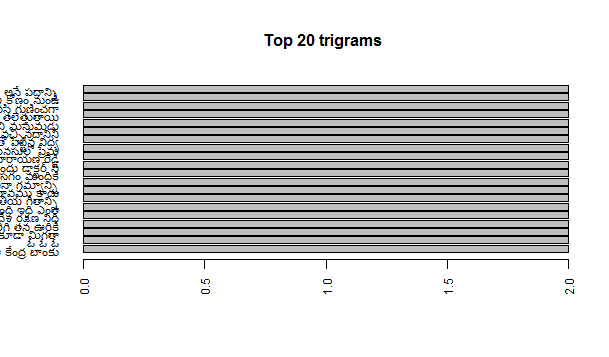
Similarly we’ll find bigrams, trigrams, quadrigram.



**Word cloud for unigrams** (maximum no. of words=100)







**17. Project Code:**

#R offers several packages to build n-grams.It was difficult to find an efficient method to construct the n-grams

#adjusted to the computing resources available. Several packages were tested and discarted, included RWeka, tm, openNLP, ngram,datasets,shiny and knitr,stringi,tm.

rm(list=ls(all=T))

libs<- c("knitr","stringi","tm","RWeka")

lapply(libs,require,character.only=TRUE )

library(knitr)

opts\_chunk$set(cache=TRUE,warning=FALSE)

freqs1<-scan(file=file.choose(), what="char", sep="\n", encoding="UTF-8")

freqs2<-scan(file=file.choose(), what="char", sep="\n", encoding="UTF-8")

freqs3<-scan(file=file.choose(), what="char", sep="\n", encoding="UTF-8")

#Media files general statistics:

#Lines

freqs1Lines <- length(freqs1)

freqs2Lines <- length(freqs2)

freqs3Lines <- length(freqs3)

#Characters

freqs1Nchar <- sum(nchar(freqs1,type="chars"))

freqs2Nchar <- sum(nchar(freqs2,type="chars"))

freqs3Nchar <- sum(nchar(freqs3,type="chars"))

#words statistics :

sum(stri\_count\_words(freqs1))

sum(stri\_count\_words(freqs2))

sum(stri\_count\_words(freqs3))

#Other statistics

#Summary statistics :

stri\_stats\_general(freqs1)

stri\_stats\_general(freqs2)

stri\_stats\_general(freqs3)

#Due to the dataset size and the processing time required,and my pc limtations i will only sample the freqs2 documents assuming that the sample size allowed by my pc is representative of the freqs2 raw data.

#Theexloratory analysis woud be similar for the other medias.

set.seed(100)

sample.size=980

mediaSample<- freqs2[sample(c(1:length(freqs2)),sample.size,replace=FALSE)]

# preview the sample before cleaning

head(freqs2)

tail(freqs2)

#text cleaningPrior to corpus Tokenization and for an adequate text mining data need to be cleaned in order to extract significant features.

The cleaning function cleans as follow:

remove no alphanumeric

remove stop word

remove URLs

remove punctuation

remove no english words

remove numbers

strip white spaces

Not all of them are necessary i will get back to this in next weeks.

doc.vec<- VectorSource(freqs2)

mycorpus<- Corpus(doc.vec)

summary(mycorpus)

mycorpus<- tm\_map(mycorpus, removePunctuation)

mycorpus<- tm\_map(mycorpus, removeNumbers)

mycorpus<- tm\_map(mycorpus, content\_transformer(tolower))

mycorpus<- tm\_map(mycorpus, stripWhitespace)

mycorpus<- Corpus(VectorSource(mycorpus ))

#Document Tokenization#

mycorpus<- tm\_map(mycorpus, PlainTextDocument)

#preview the sample after cleaning

head(mycorpus)

tail(mycorpus)

#Tokenization :ngram analysis

#Tokenize the corpus into n-grams:

#We need to convert our cleaned mediaSample (cmediaSample) to corpus for

#further exploration like ngrams(1,2,3) frequency distribution using packages tm and RWeka

# library

corp<- tm\_map(corp,stemDocument)

#3 ngrams tokenization functions for unigrams,bigrams and trigrams

#Set the default number of threds to use, it is needed for ngram function.

options(mc.cores=1)

uniTokenizer<- function(t) NGramTokenizer(t,Weka\_control(min=1,max=1))

biTokenizer<- function(t) NGramTokenizer(t,Weka\_control(min=2,max=2))

triTokenizer<- function(t) NGramTokenizer(t,Weka\_control(min=3,max=3))

quadTokenizer<- function(t) NGramTokenizer(t,Weka\_control(min=4,max=4))

#Document TERM Matrix

#Create Document Term Matrix and calculate frequencies using package tm and plot the 20 most frequent ngrams.

#Top 20 most frequent unigrams

dtm<- DocumentTermMatrix(mycorpus, control=list(tokenize=uniTokenizer))

unifreq<- colSums(as.matrix(dtm))

barplot(tail(sort(unifreq),20),las=2,main="Top 20 unigrams",cex.main=1,cex.axis =0.75,horiz=TRUE)

#Top 20 most frequent bigrams

dtm<- DocumentTermMatrix(mycorpus,control=list(tokenize=biTokenizer))

bifreq<- colSums(as.matrix(dtm))

barplot(tail(sort(bifreq),20),las=2,main="Top 20 bigrams",cex.main=1,cex.axis

=0.75,horiz=TRUE)

#Top 20 most frequent trigrams

dtm<- DocumentTermMatrix(mycorpus,control=list(tokenize=triTokenizer))

trifreq<- colSums(as.matrix(dtm))

barplot(tail(sort(trifreq),20),las=2,main="Top 20 trigrams",cex.main=1,cex.axis

=0.75,horiz=TRUE)

Top 20 most frequent quadrigrams

dtm<- DocumentTermMatrix(mycorpus,control=list(tokenize=triTokenizer))

quadfreq<- colSums(as.matrix(dtm))

barplot(tail(sort(quadfreq),20),las=2,main="Top 20 quadrigrams",cex.main=1,cex.axis=0.75,horiz=TRUE)

#Word Cloud

library(wordcloud)

freq.onegram<- sort(rowSums(as.matrix(unifreq)), decreasing = FALSE)

wordcloud(names(freq.onegram), freq.onegram, max.words = 300, colors = brewer.pal(6, "Dark2"))

freq.bigram<- sort(rowSums(as.matrix(bifreq)), decreasing = FALSE)

wordcloud(names(freq.onegram), freq.onegram, max.words = 300, colors = brewer.pal(6, "Dark2"))

freq.trigram<- sort(rowSums(as.matrix(trifreq)), decreasing = FALSE)

wordcloud(names(freq.onegram), freq.onegram, max.words = 300, colors = brewer.pal(6, "Dark2"))

**18. CONCLUSION AND NEXT STEPS:**

* I was really limited by my computer 2G RAM ,so i had to reduce the corpus size to the minimum possible. This issue should be re-examined ,may be more RAM or an online text exploration ,to be able to use a more data representative sample ,and do an adequate stemming and parsing ,also use a dictionary that can be passed to the DocumentTermMatrix as a control argument.
* I’m planning to build a bigger corpus from the data, sampling at least the 40% of the total data.
* The use of well designed functions may definitely help.
* how to take into account smoothing when test set is not all covered by training set.
* I expect to use n-gram models up to 4-gram , with a back off strategy
* This code applicable to all languages(except English)
* Next step is design Machine learning algorithm for word prediction based on this files.

**19. References** :

* Quantitative corpus linguistics with R: a practical introduction.
* Data Science Milestone Projects by apikouAbdelkaderESSODEGUI(for English text files):From Rpubs.
* Dr.RahulBalusu case studies materials(special thanks).