# Sentence Boundary Disambiguation: A User Friendly Approach

Pritam Singh Negi
Department of Computer Science
H.N.B. Garhwal University
Srinagar Garhwal, India

M.M.S. Rauthan
Department of Computer Science
H.N.B. Garhwal University
Srinagar Garhwal, India

H.S. Dhami
Head, Department of Mathematics
Kumoan University, Nainital

## ABSTRACT

In the present work we have developed an algorithm based on maximum entropy and stop word removal modules, which works with almost 99% accuracy and have established supremacy over the existing paragraph breaker software developed by Text Mining Group, School of Computer Science, Manchester University, United Kingdom .

**Keywords**: Sentence Boundary, Information retrieval, Evaluation.

## 1. INTRODUCTION

Sentence Boundary Disambiguation (SBD) has received increased attention in recent years as a way to enrich speech recognition output for better readability and improved demonstrations in many applications of Natural Language Processing, like: Parsing, Information Extraction, Machine Translation, POS tagging and Document Summarization. Among the most relevant works, we can cite the names of Berger (1996), Palmer & Hearst (1997), Mikheev (2000), Manning & Schutze (2002), Kiss & Strunk (2006), Xuan et al (2007), Siminski (2007) and Gillick (2009) etc. to mention only a few.

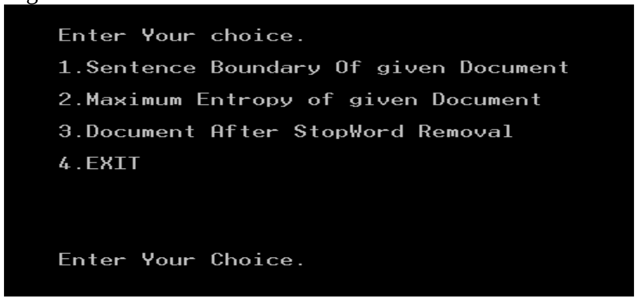
We know that sentence is a sequence of words ending with a terminal punctuation, such as '.','?', '!' etc. Most sentences use a period at the end. However, sometimes a period can be associated with an abbreviation, such as "Mr. or mr, U.S.A., Ph. D., M. Sc. etc." or can represent a decimal point in a number like 102.53. In all these cases, it is a part of an abbreviation or a number. We cannot delimit a sentence because the period has a different meaning here and therefore there arises an ambiguity in breaking the sentence. To establish the task of sentence boundary disambiguation for a given document there are certain necessary conditions those are very important when breaking a sentence boundary disambiguation.

In this paper we have made an attempt to provide a system which can be implemented in any system and can deduce the sentence boundary with high accuracy. For this purpose, we have considered the following conditions through which our system provides the high accuracy for detecting the sentence boundary:

- not to break a sentence when the sentence contains certain abbreviated words like *I.B.M., Ph. D. etc*.
- not to break a sentence when the sentence contains the numeric words in it like "My percentage in post graduation is *85.00%*."
- not to break a sentence when multiple occurrence of a sentence terminator in a single word likes *www.google.com.*
- take care of the sentences which contains multiple occurrence of a particular sentence breaking character (like *?,.,!)*.
- take care of the sentences which are most common and where a sentence should not be broken like *Mr., Mrs., Col.*
- take care of sentence with the words like etc., *e.g., i.e.*

## 2. LOGIC AND ALGORITHM

Logic, flow and description of various important algorithms/ functions used in the program and related snapshots for sentence boundary detection have been given in this paper along with the calculation of the maximum entropy of the given document. The logic is solely implemented in C language and performs well with English text.



```
Enter Your choice.
1.Sentence Boundary Of given Document
2.Maximum Entropy of given Document
3.Document After StopWord Removal
4.EXIT


Enter Your Choice.
```

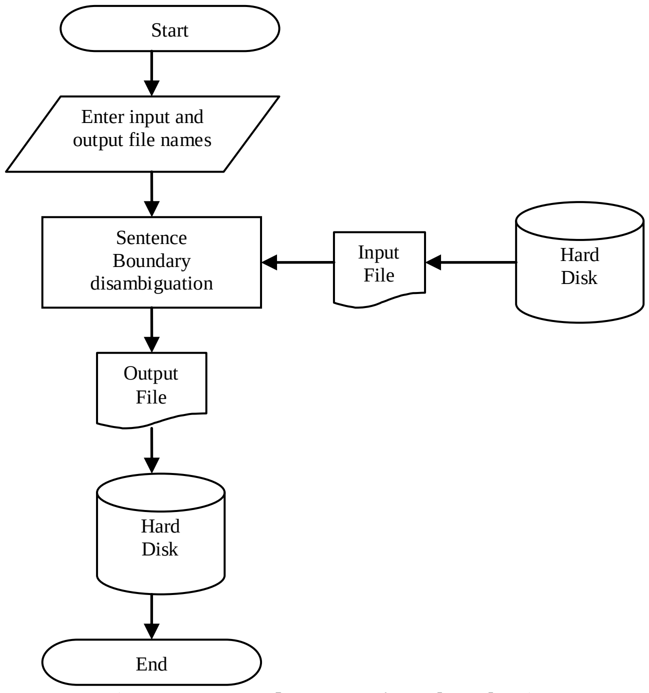
***(Main Module)***

## 2.1 SENTENCE BOUNDARY DETECTION

The system takes a Text File with English Text which may contain Alphanumeric Text. First of all, we check the certain frequent words like (*mr., mrs*.). For this to happen, process the text file character by character and create a whole word from that to write it to the output file (final sentenced document), match this word (before writing it to another file which one is a actual output file) with the array which contains frequent occurring words. If the word matches any of array elements we write the word to the file without writing any newline character to the file.

Afterwards the system checks the digits that may arrive in a document. For this to happen, process the file character by character and as soon as encounter a digit store it in a temporary buffer and match its next character if it is a "." Character then store this character in the buffer too and match the next character if it is a space then surely the buffer does not contain a floating point number and place it to the file without writing any newline character to the file and if there is any digit after the "." place the digits to the buffer till next "." or "?" or"!" is not arrived and finally places the contents of buffer to the file.

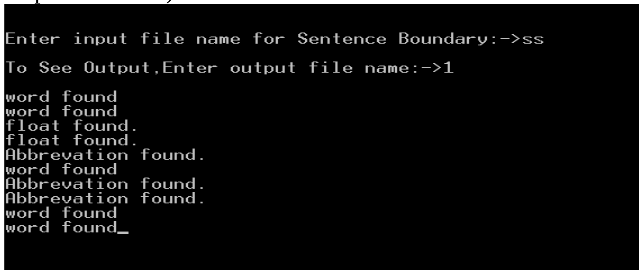
Now check the word, which contains the words like www.google.com. For removing this type of problem first process the file character by character and then as soon as the character 'w' is encountered, stored it in a buffer and checks for next character, if it is again 'w' then store it too in the buffer and check for another one, if finally get the word 'www.' traverse the file contents to the point till the " "(space) character is not encountered (because www.google.com) does not have any spaces in between, and store all contents in a buffer and then finally to a file. Now check other additional tasks like multiple occurrence of a sentence terminator to the sentence like (Ohh My God!!!!!!!!!!!!!). Here, system does not break a sentence in the first sentence terminator (!) so copy all the terminators in the buffer and then store them in the file.


Start
Enter input and output file names
Sentence Boundary disambiguation
Input File
Hard Disk
Output File
Hard Disk
End

***(Sentence Boundary Detection Flow chart)***

Input: A text file (a document with English text).
Output: An output file (a document with proper sentences.)
Steps: There are following steps:
Step 1: take an array preword[] to store most frequent names prefixes like Mrs., Mr., col., Dr., Lt., Ph. D., A. D. , B. C. etc.

Open a document in read mode (say input document) for reading and a document in writing mode for writing final output (say output document).



```
Enter input file name for Sentence Boundary:->ss

To See Output,Enter output file name:->1

word found
word found
float found.
float found.
Abbrevation found.
word found
Abbrevation found.
Abbrevation found.
word found
word found_
```

***(Sentence Boundary Detection Module)***

```
Step 2:    while(End of document is not reached)
           Do
                Process the document
                character by character
           Now check the following:
```

Step 2a: Check for Sentences Containing Abbreviations

If the word document contains any abbreviations like G.W.Bush then we check that if these sentences start with uppercase letter and follow by a '.' then continue this process till the space character is encountered. This word is not part of sentence boundary and is written to the file without writing any newline character to the file.

Step2b: Check for sentence containing any word prefixes. If the word contains any digit process the document character by character and if any digit is found in a word then we store it in a temporary buffer and process next character. If this character is a "." then store this in temporary buffer and proceed next. If it again is a digit then through a loop traverse file till a space character is encountered and store it in temporary buffer and do not place any newline character to the file.

```
c= getchar(frp);
int i=0;
if(isdigit(c))
    Tempbuf[i++]=c;
    Proceed to next char
    C=getc(frp);
If this char is a ".", store it in tempbuf.
    Tempbuf[i++]=c;
    And then check next char.
    C=fget(frp);
    If (c!='.')
        Then the tempbuf does not contain any
        floating-point digit and check for the case
        (uppercase or lowercase for c), if c is of type
        upper then place a newline char to the file.
    Else
        write the content of tempbuf to the file
        without writing any newline character to the
        file.
   Else
    Then the tempbuf contain floating point digit. And
    we check for the next space character by traversing
    through a while loop:
While(c!=' ')
Do
        Tempbuf[i++]=c; and then place the content
        of tempbuf to the file without writing any
        newline character to the file.
EndDo
}
Store each character on a buffer till a whole word is not
formed. Say this word is stored in a temporary
buffer(tempbuf). If this word (tempbuf) matches with
the word contained in preword[ ] array.
If(!strcmp(tempbuf,preword))
Then
        This word is not a part of sentence boundary
        and writes it to the file without writing any
        newline character to the file.
fwrite(tempbuf,outputdocument);
Else
        This word is a part of sentence boundary and
        places a newline character to the output
        document.
EndElse
Endif
```

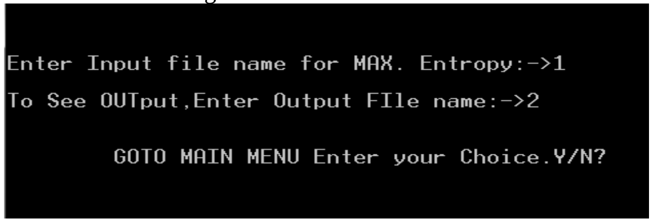
Step2.c. Check for the sentences containing the word like www.google.com. For this traverse the file character by character and check as soon as the character 'w' or 'W' is encountered.

```
        If(c=='w'||c=='W')
        Then store the char to the buffer and proceed next
        Tempbuf[i++]=c;
        C=fget(frp);
If it is again a 'w' or 'W' then store the char to the buffer and proceed next
        Tempbuf[i++]=c;
        C=fget(frp);
If it is again a 'w' or 'W' then store the char to the buffer and proceed next
        Tempbuf[i++]=c;
        C=fget(frp);
If it is a '.' Then we have encountered a word www. And we proceed through till we encounter any space character.
        While(c!=' ')
        do
        Tempbuf[i++]=c;
        EndDo
And write final output to the file without writing any newline character to the file.
EndDo
```

This is the algorithm for sentence boundary disambiguation algorithm and work with an accuracy of 99% approx. The 1% erroneous results are due to the versatility of the English language and due to the use of words in different manner by different peoples while writing their documents. For example some people may proceed their name only by Mr instead of Mr. So a problem may arise by sentence boundary algorithm here.

## 2.2 MAXIMUM ENTROPY

For finding more and more accurate result our implementation of sentence boundary disambiguation algorithm uses the mathematical model known as Maximum Entropy Model. It finds the total occurrences of periods, question marks and exclamation marks and also finds actual sentence breaking occurrences of periods, question marks and exclamation marks. Then it applies some mathematical formulae to produce a result known as Entropy of the sentence. This help in improving the precision and recall of the searching of the document.



```
Enter Input file name for MAX. Entropy:->1
To See OUTput,Enter Output FIle name:->2

        GOTO MAIN MENU Enter your Choice.Y/N?
```

***(Maximum Entropy Module)***

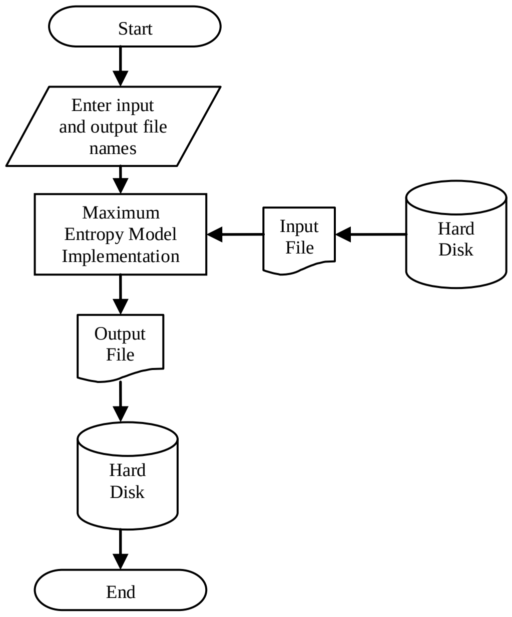
In this module we take an input document as input which is preprocessed by sentence boundary module and check for the occurrence of total number of dots, actual sentence breaking dots, question marks and actual sentence breaking question marks, total exclamation mark and actual exclamation marks in the given document and then find the entropy for each (.,?,!) in the document and save the result in the another document.

Enter the name of the file where the text is saved which comes after running the sentence boundary algorithm to the actual document.

Enter the file name where the output will be saved and open this file into write mode.

Now check how many question marks ('?') available on the document and how many question mark define the actual sentence boundary.

Similarly check it for '.' and exclamation ('!') marks.


Start
Enter input and output file names
Maximum Entropy Model Implementation
Input File
Hard Disk
Output File
Hard Disk
End

***(Maximum Entropy Flow chart)***

Now find the probability of the Sentences that end with dot with the help of following formulae:

dot_prob(SB)+dot_prob(NSB)=1

```
if(ad>0&&cd>0) where cd= Total Dots and
                     ad= Actual Sentence Boundary In Dots
{
    dot_prob_for_SB = ad/cd;
    dot_prob_for_NSB = (1.0-dot_prob_for_SB);
}
```

Then calculating the entropy of dot sentence boundary with the help of formulae H(P)=-$\sum_{i=1}^{n} pi \log i$

Entropy_for_dot
= - ((dot_prob_for_SB* log(dot_prob_for_SB)) + (dot_prob_for_NSB*log(dot_prob_for_NSB))

Probability of Dot as Sentence Boundary = dot_prob_for_SB

Probability of Dot as not a Sentence Boundary = dot_prob_for_NSB

Entropy for Dot = entropy_for_dot

Similarly found the probablity of the Sentences that end with question mark.

```
if(aq>0 && cq>0)   where aq= Total question marks and
                         cq= Actual question marks
{
    ques_prob_for_SB=aq/cq;
    ques_prob_for_NSB=(1.0-ques_prob_for_SB);
}
```

Then calculating the entropy of question mark sentence boundary with the help of formulae H(P)=-$\sum_{i=1}^{n} pi \log i$

Entropy_for_ques
=-((ques_prob_for_SB*log(ques_prob_for_SB))
+(ques_prob_for_NSB*log(ques_prob_for_NSB))

Probablity of Question Marks as Sentence Boundary

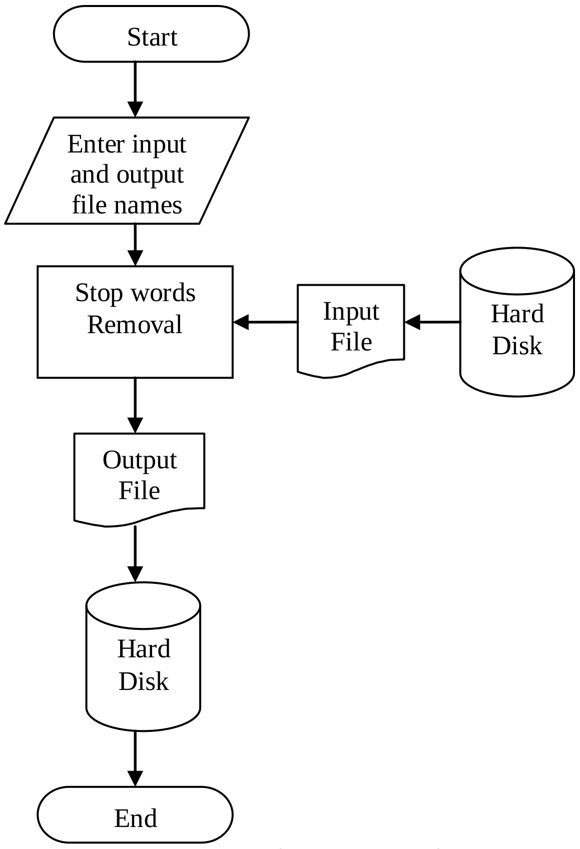
= ques_prob_for_SB
Probablity of Question Marks not as a Sentence Boundary =ques_prob_for_NSB
Now found the probability of the Sentences that end with exclamation mark. The probability for SB and NSB will be same because exclamation mark may be sentence punctuation or not.

```
if(ae>0&&ce>0) where ae= Total exclamation marks
                         and ce= Actual exclamation marks
{
        excl_prob_for_SB=ae/ce;
        excl_prob_for_NSB=(1.0f-excl_prob_for_SB);
}
```

Then calculating the entropy of exclamation mark sentence boundary with the help of formulae H(P)=-$\sum_{i=1}^{n} \mathrm{pi} \log i$
Entropy_for_excl
=-((excl_prob_for_SB*log(excl_prob_for_SB))
+(excl_prob_for_NSB*log(excl_prob_for_NSB))
Probability of Exclamation Mark as Sentence Boundary
= excl_prob_for_SB
Probability of Exclamation Dot as not a Sentence Boundary
= excl_prob_for_NSB
Entropy For Exclamation Mark = entropy_for_excl

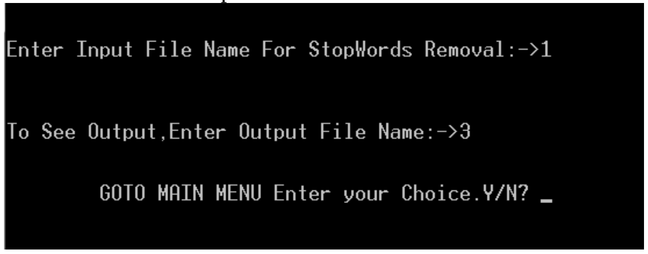
## 2.3 STOP WORD REMOVAL

This module has been used to carry certain most frequently used words like (“a”, ”the”, ”an”, ”are” etc.), since a search using one of these terms is likely to retrieve almost every item in the database regardless of its relevance, so their discrimination value is low.



*(Removing stop words from document flow chart)*

Furthermore, these words make up a large fraction of the text of most documents, the ten most frequently occurring words in English typically account for 20 to 30 percent of the tokens in a document. Neglecting such words from consideration early in automatic indexing speed processing, saves huge amounts of space in indexes and does not damage retrieval effectiveness. A list of words filtered out during automatic indexing is called stopword list or a negative directory. Our algorithm successfully removes about 512 stopwords stored in a file.



```
Enter Input File Name For StopWords Removal:->1


To See Output,Enter Output File Name:->3

        GOTO MAIN MENU Enter your Choice.Y/N? _
```

***(Stop Word Removal Module)***

Enter the name of the file For StopWords Removal.
Enter the file name where the output will be saved and open this file into write mode.
Create a file in which all the stopwords are present.
Now compare stop words one by one to whole document.
If stop words found then remove it from the actual document and store it to output document.

In the stopwords removal module, we have taken only 512 stopwords, while there are more than 700 stopwords and we can easily add remaining stopwords into the program.

## 3. COMPARISON

When we compare our system with the sentence and paragraph breaker software, which one has been developed by Scott Piao member of Text Mining Group, School of Computer Science, Manchester University, UK., we observe as under:

- Test mining group software has been developed in JAVA language and our system is developed in C language.
- Test mining group software is available online and one can work with it only when the Internet is working on that computer where the software is working. Our system can also be developed for Internet with more accuracy.
- Previous software can break the sentences on most common uses word like mr., mrs. etc which one is not an actual sentence breaking position but our system can break the sentence in its actual position.
- Previous software cannot break the sentences when only one character is found before the “.” in place of a word but our system can recognize the end position and break the sentence in its actual position.

## 4. CONCLUSION

We have presented a system for sentence boundary detection of English text. In this system we have attempted to create an open source software tool and the experimental results show that the approach can achieve a high accuracy. With the help of this system anyone can detect sentence boundary of a given text

document. This work shall have applications in Information Retrieval System because without breaking a text document into sentences, there is no meaning of applying text operations on the document, and therefore we cannot retrieve relevant (according to query) information from the document.
This work removes all the problems that occur in sentence boundary disambiguation. It finds actual sentence ending for the given text documents paragraphs.

## 5. REFERENCES:

[1] Berger A. 1996. A Brief Maxent Tutorial. http://www-2.cs.cmu.edu/~aberger/maxent.html.

[2] Gillick Dan (2009) Sentence boundary detection and the problem with the U.S.(2009) Proceeding of the NAACL HLT2009:Association for computational linguistics, (short papers),241-244.Boulder,Colorado.

[3] Kiss T. and Strunk J.(2006) Unsupervised multilingual sentence boundary detection, Computational linguistics, 32(4), 485-525.

[4] Manning, C.D. and Schütze, H. (2002) Foundations of statistical natural language processing. The MIT Press, Cambridge/London.

[5] Mikheev, A. (2000). Tagging Sentence Boundaries. In Proceedings of the NAACL, pp 264-271, Seattle, WA.

[6] Palmer, D.D. & Hearst, M.A. (1997). Adaptive Multilingual Sentence Boundary Disambiguation. Computation Linguistics, 23(2), 241-269.

[7] Siminski Krzysztof (2007) Sentence boundary verification in Polish text, Computer recognition systems 2, Advances in soft computing, Springer, Vol.45/2007,493-499.

[8] Weijian, Xuan, Watson, Stanley J. and Meng Fan (2007) Tagging sentence bares to Biomedical literature, Computational linguistics and Intelligent text processing, Lect. Notes in Computer Science, Springer, No.7, Vol.4394/2007, 186-195.