**PROJECT REPORT**

on

Auto Text Summarization

Submitted By

**Nachiket Prakash Tembhurnikar**

UE173065

**Sakshi Chandel**

UE173087

**Saloni Mohan**

UE173088

**Shivam Batra**

UE173095

**Surjit Kumar**

UE17108

**Visarika Vaidya**

UE173119

Submitted To

**Mrs. Dhriti**

**Mrs. Sunaina Gulati**

****

**Computer Science and Engineering**

**University Institute of Engineering and Technology**

**Panjab University, Chandigarh-160014, INDIA**

**2018**

**Table of Content**

|  |  |
| --- | --- |
| Topic | Page No. |
| 1.Introduction | 1-4 |
| 2.Hardware and Software Requirements | 5 |
| 3.Work Flow | 6-12 |
| 4.Screenshots | 13-19 |
| 5.Future Scope | 20 |
| 6.Conclusion | 21 |
| 7.References | 22 |

1. INTRODUCTION

There has been an exponential growth in the amount of information available on the Web. The web is filled with a lot of books, papers, journals and text HTML pages, all of which contain useful information for an intending researcher. However, the difficulty arises when it comes down to getting the relevant information to his area of research. A person cannot possibly go through all of the available materials online for a study – One might finish a paper and realize it didn’t contain what he needed in the first place, a considerable waste of time! What he will probably need is a summarized version of that text which will give a short description of the whole body of text thus saving the person valuable time and resources. Also, “due to the explosive amounts of text data being created and organizations increased desire to leverage their data corpora, especially with the availability of Big Data platforms, there is not usually enough time to read and understand each document and make decisions based on document contents” (Elfayoumy & Thopil, 2014). Hence, there is a great demand for summarizing text documents to provide a representative substitute for the original documents. Hence, the need for a text summarizer.

The area of Text Summarization has been researched extensively over the last two decades and algorithms and techniques have been proposed to achieve the purpose of summarizing whole text documents. Text summarization aims to produce a shorter representation of the input text. It is quite useful in finding out whether or not a lengthy document is worth further reading by the user. This is achieved by reducing the length of the main text while maintaining its main point and overall meaning. An excerpt from the ROUGE website, berouge.com about says it all the direction of research into text summarization. It is given below:

“*Automated text summarization has drawn a lot of interest in the natural language processing and information retrieval communities in the recent years. A series of workshops on automatic text summarization (WAS 2000, 2001, 2002), special topic sessions in ACL, COLING, and SIGIR, and government sponsored evaluation efforts in the United States (DUC 2002) and Japan (Fukusima and Okumura 2001) have advanced the technology and produced a couple of experimental online systems* (Radev, et al., 2002)*. Despite these efforts, however, there are no common, convenient, and repeatable evaluation methods that can be easily applied to support system development and just-in-time comparison among different summarization methods”.*

**Text summarization** is a reductive transformation of source text to summary text through content selection and/or generalization on what is important in the source. Nowadays with the exponential growth of the information available through internet, in other words, the information overload, the volume of the

information is much more than the need of people. Certainly, studying this massive volume of information would be tedious and time consuming. Meanwhile, without studying a document, one cannot understand its contents and decide about its relevance with the topic. Because of this, having an alternative approach that can help reader achieve the main points without reading the whole text, so he can decide whether or not to read the whole text, would be really beneficial.

Different categorizations, based on different points of view, have been suggested for automatic text summarization systems (for more study). Based on their output, summarization systems are categorized as either ‘*extractive’* or ‘*abstractive’*. In extractive systems the output contains the most important parts (sentences, paragraphs, etc.) of the original text, without making any changes to them; while in abstractive systems, the summarization is done through understanding the original text, then retelling it using fewer words. Also based on their output, text summarization systems are categorized as ‘*generic’* or ‘*query-based’*. Output of the generic summarization systems contains the most important points of the given document, while in query-based systems, the summary only contains the concepts that are closely related to the query.

Perhaps the rules for correct composition of paragraphs that in 19th century Alexander Bain have suggested in his English Composition and Rhetoric (1866), is the beginning of taking coherence in text seriously. This concept, which garnered much more attention later, has two ‘*local*’ and ‘*global*’ kinds. The local coherence is semantic relationship between each pair of consecutive sentences in text (sentence to sentence transactions); while the global coherence is the semantic relation that exists between the all sentences. Of course, the local coherence is a necessary prerequisite for global coherence, to the extent that McKoon and Ratsliff say psycholinguisticaly it is the most important source for inference-making during the study/listening. Hence, ‘local coherence’ (hereafter ‘coherence’) has been subject to more attention in computational linguistics.

Lack of coherence in text or speech will most likely prevent the message from being delivered to the reader or listener. That is why coherence of the output text is one of the key aspects of any text producing systems. Distribution of entities in locally coherent texts, displays certain regularities. Therefore, ontology as a knowledgebase of entities and their semantic relations, can be a suitable basis for achieving this regularity and consequently, coherence. “Ontology” is a Greek word consisting of “*onto*” meaning “beings” and “*logos*” which usually is interpreted as “science”, therefore it can be said that ontology is science or study of beings. In other words, ontology is a systematic account of existence that is used for modelling the beings in real world and the relations between them.

Usually, based on the nature of query-based text summarization (only sentences closely related to the query are put in the summary), their output has more coherence than summaries produced using generic text summarization systems (which put sentences related to all the topics made in the text, in the summary). In other words, in generic summarization, because of the variety of topics there is less semantic relation between sentences. The goal of this paper is to devise and implementation of an extractive and generic text summarization system for documents that produces more coherent summaries. Therefore, without any queries, system will omit less important topics and produce a summary consisting of the most relevant sentences to the main topic discussed in the original document, which has a high level of semantic relation between its sentences. In many of the applications, without any queries, there is need to get a summary of the most important/central topic of the document, in a way that sentences have semantic relation. For this purpose, a system will be designed that ontology is the basis for forming the summaries (ontology-based automatic text summarization), also the concepts extracted from it will be used for producing more coherent summaries. In this regard, the second part is dedicated to studying the related works done in this field, and in the third part the suggested automatic text summarization system and its approach toward achieving coherent summaries are discussed. Then in the fourth part, discussing the evaluation methods of automatic text summarization systems, the results will be evaluated, and finally in the fifth part, conclusion of this research will be presented.

1. HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements

* Win 95/98/XP with MS Office and JDK 8 or higher version

Hardware Requirements

|  |  |
| --- | --- |
| **Computer and processor** | 1 GHZ or faster x86 or 64-bit processor with SSE2 instruction set |
|  | 1 GB RAM (32-bit) |
| **Memory** | 2 GB RAM (64-bit) recommended for graphics features, Outlook Instant Search & certain advanced functionality |
| **Disk space** | 3 gigabytes (GB) |

1. WORK FLOW
2. **Extract Sentences from Context ()**- It stores each sentence in an Array List(Sentence)

Read file

If br.read()!=-1

True False

Store character in temp[]

False

If br.read()=’.’

True

Store temp[] in Array List Sentence, temp[]=Null

1. **get freq ()**- It stores frequency of each word in an Array List(Word) using Hash Map.

Create string, integer Map

Replace , . ? etc. with space in a string that stores file i.e allwords

Split words of allwords and store in arrw[]

Int i=0

Store key and value in ArrayList Word

i<arrw.length

False i++

True

Insert key,value=1

arrw[i] is already a key.

False

True

Increase value of the key by 1

1. **tfid ()** - It finds TFIDF i.e. Term Frequency inverse Document Frequency

False

Word vb : myword

vd.tfid=vb.tf\*(Math.log(1/vb.df))

End of tfid()

1. **get df ()**- It stores document frequency of each word appearing in the file in an Array List(Word).

key[] to store file in String ,line[] to read line

Store files a,b,c,d and e in an array

int i=0

If i<5

False i++

True

Store file in key [i]

i=0

if i<5

False i++

True

word.value=word in key [i]

k.df=k.df+1

k.df=k.df/6

1. **sort ()**- It sorts number, value and score of group of sentences

Sentences Array List

i=initial index for sorting

n =no. of executions

int j=0

if

J<n

j-loop ends

j++ False

True

int k=i

if

k<i+n-j

False k++

True

Score at k< score at k+1

k>=last index of sentences

Swap score,value and number at index k and k+1

k-loop ends

1. **output ()**- It finds the score of each sentence and stores in an Array List(Sentence) and then prints the summary of the document.

False

Sentence sen: sentences

True

Split words and store in array sword ,wi=0

int j

If

j<sword.length

False j++

sen.score= sen.score/wi

True

False

Word k: myword

True

If

sword[j]= k.value

True

sen.score= sen.score+k.tfidf, wi++

Print Summary

ss=-1,lsen=last sentence number, p=0, sman=0

arrsum[p]=0,p++

i=0

If

i<sentences.size()-1

False

True

sort()

smax> score at i

False

i+=5 True

smax=score at i, arrsum[p]=ss, p++, ss=number at i

arrsum[p]=number at i, p++, smax= score at i+1, ss= number at i+1

arrsum[p-1]= lsen

False

True

arrsum[p]=lsen

Sorting of arrsum[]

Print arrsum[]

1. **main ()**- It creates an object and calls all the methods.

b1.tfid()

b1.getdf()

b1.test()

b1.extractSentenceFromContext()

bi.init()

Create object b1

throws IOexception

b1.output()

1. SCREENSHOTS

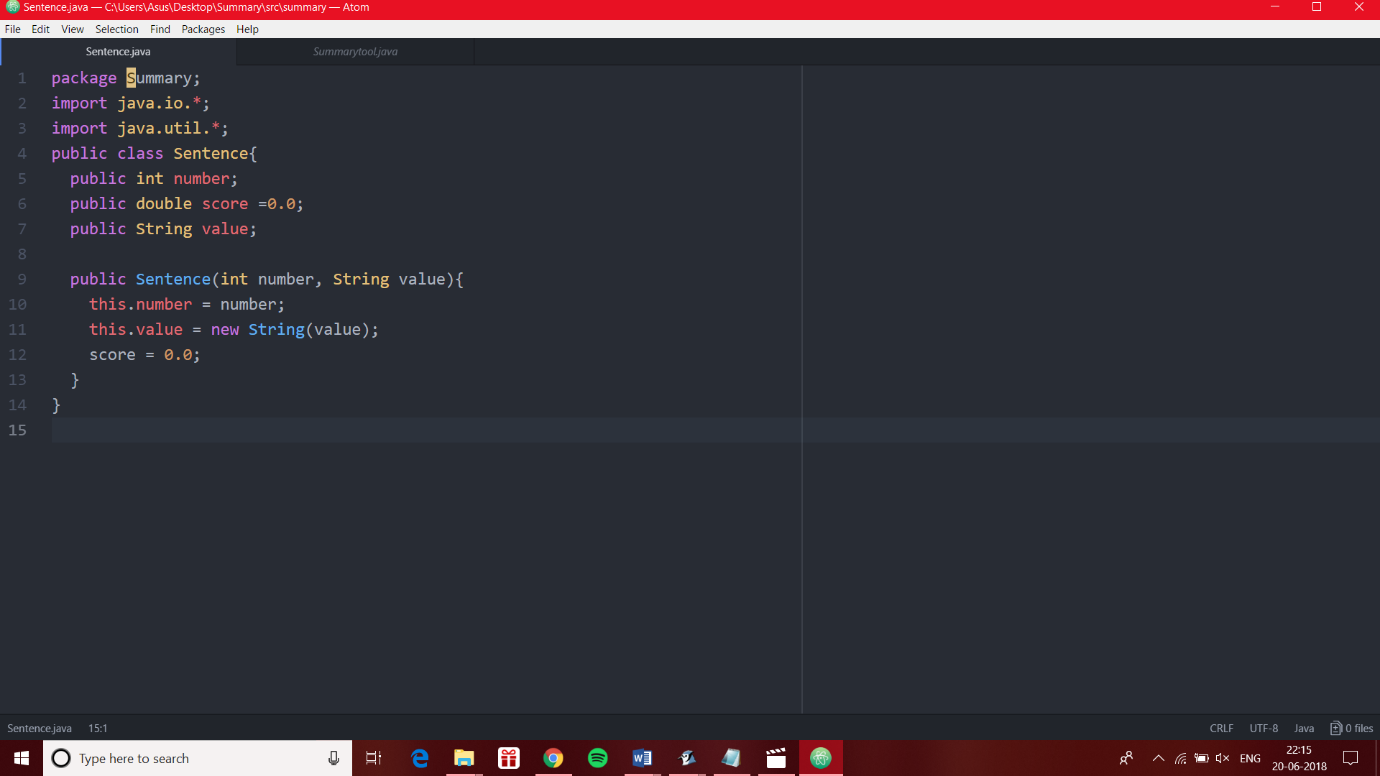


Fig.1: User-defined Array List (Sentences)

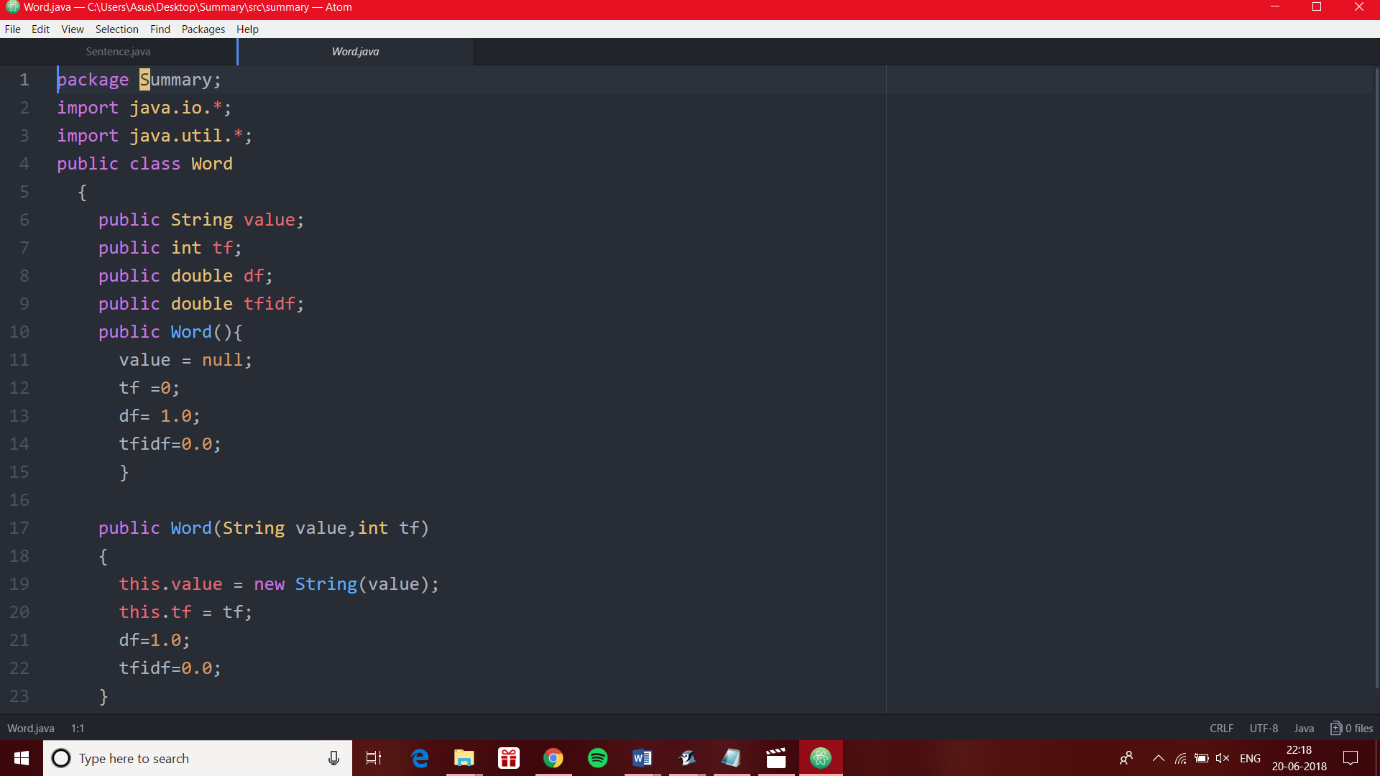


Fig.2: User-defined Array List (Word)

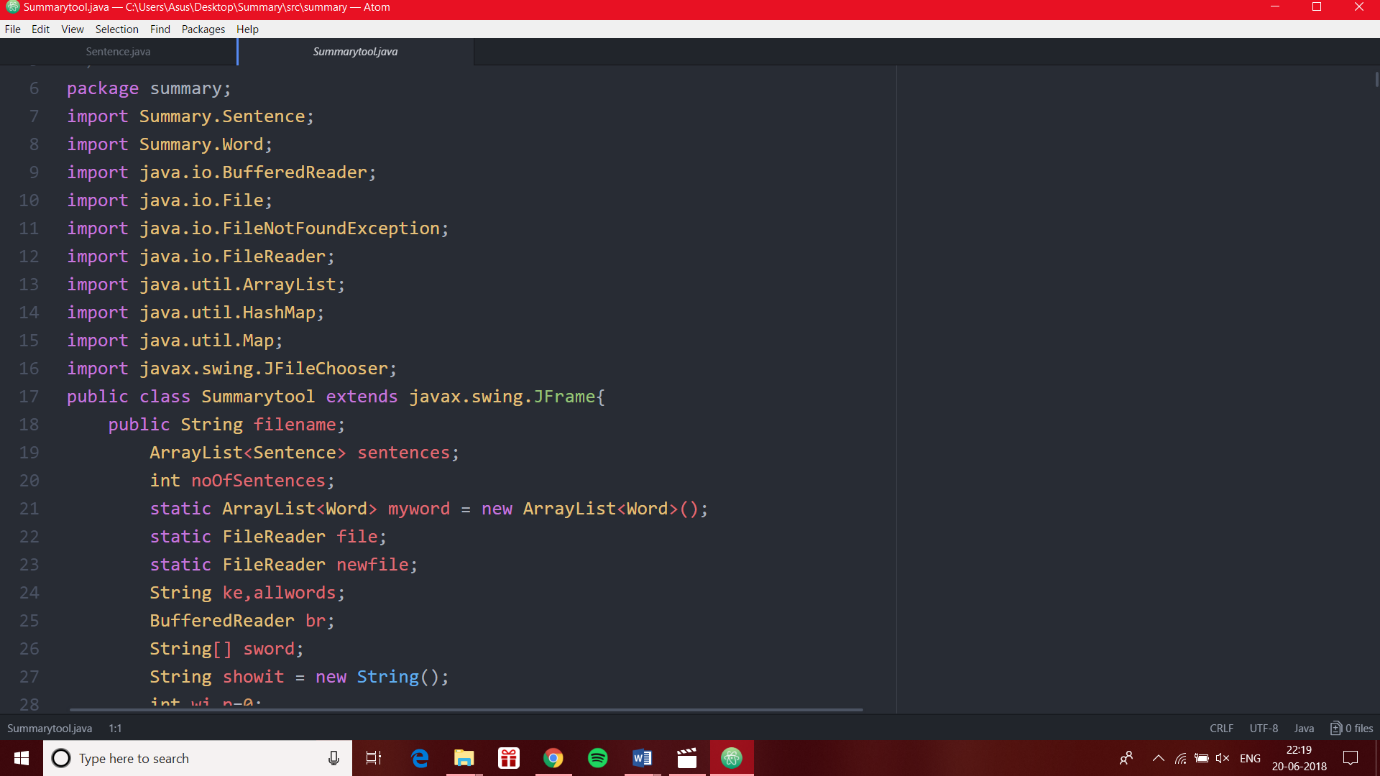


Fig.3: class Summary

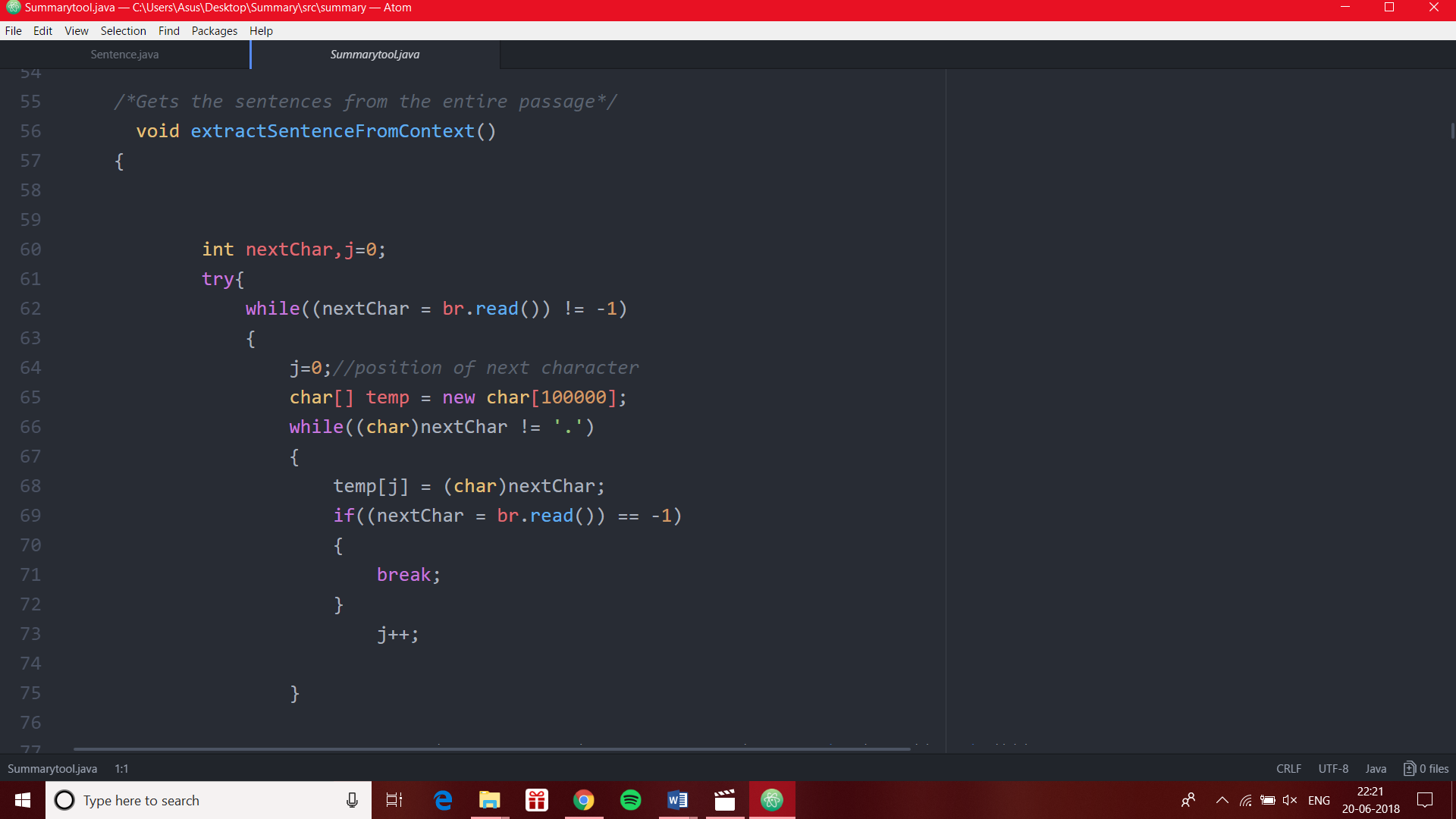


Fig.4: extract Sentence From Context ( ) method

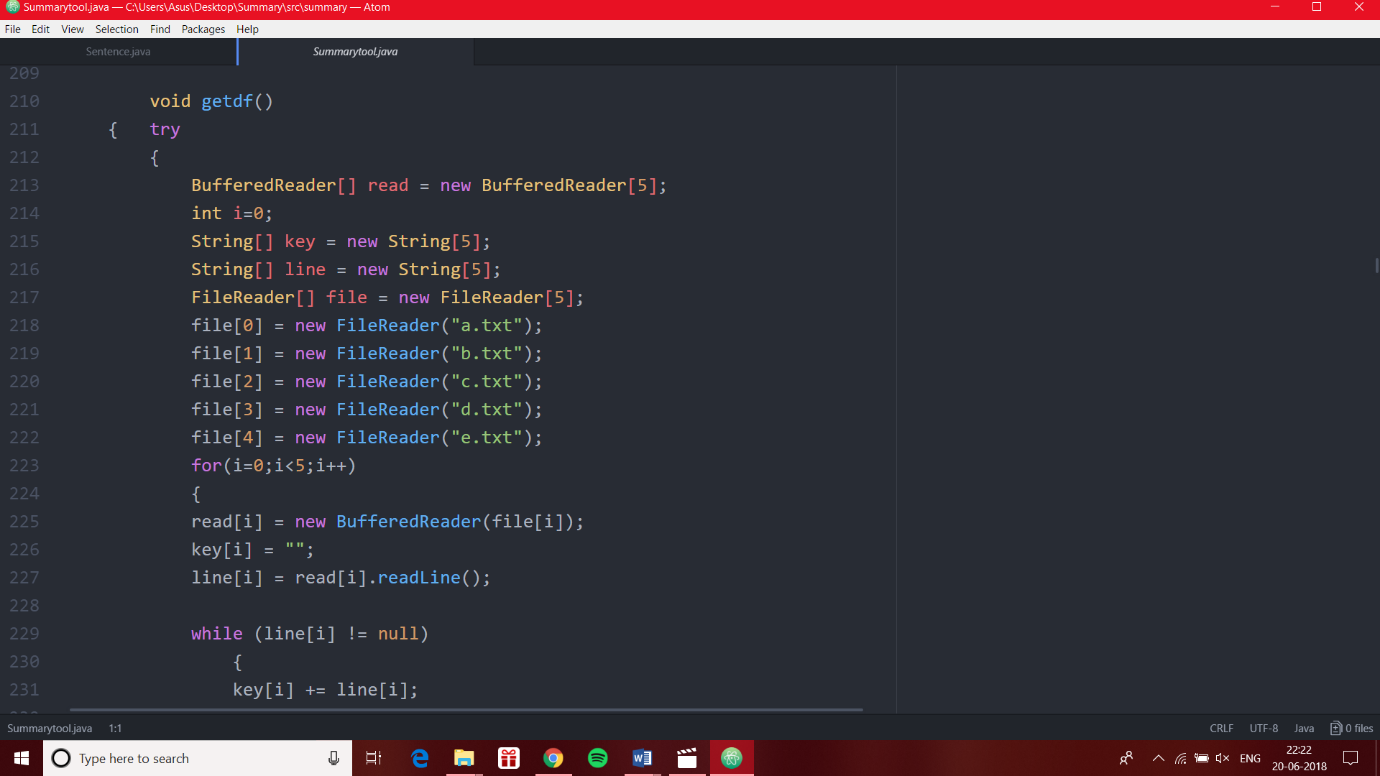


Fig.5: getdf ( ) method

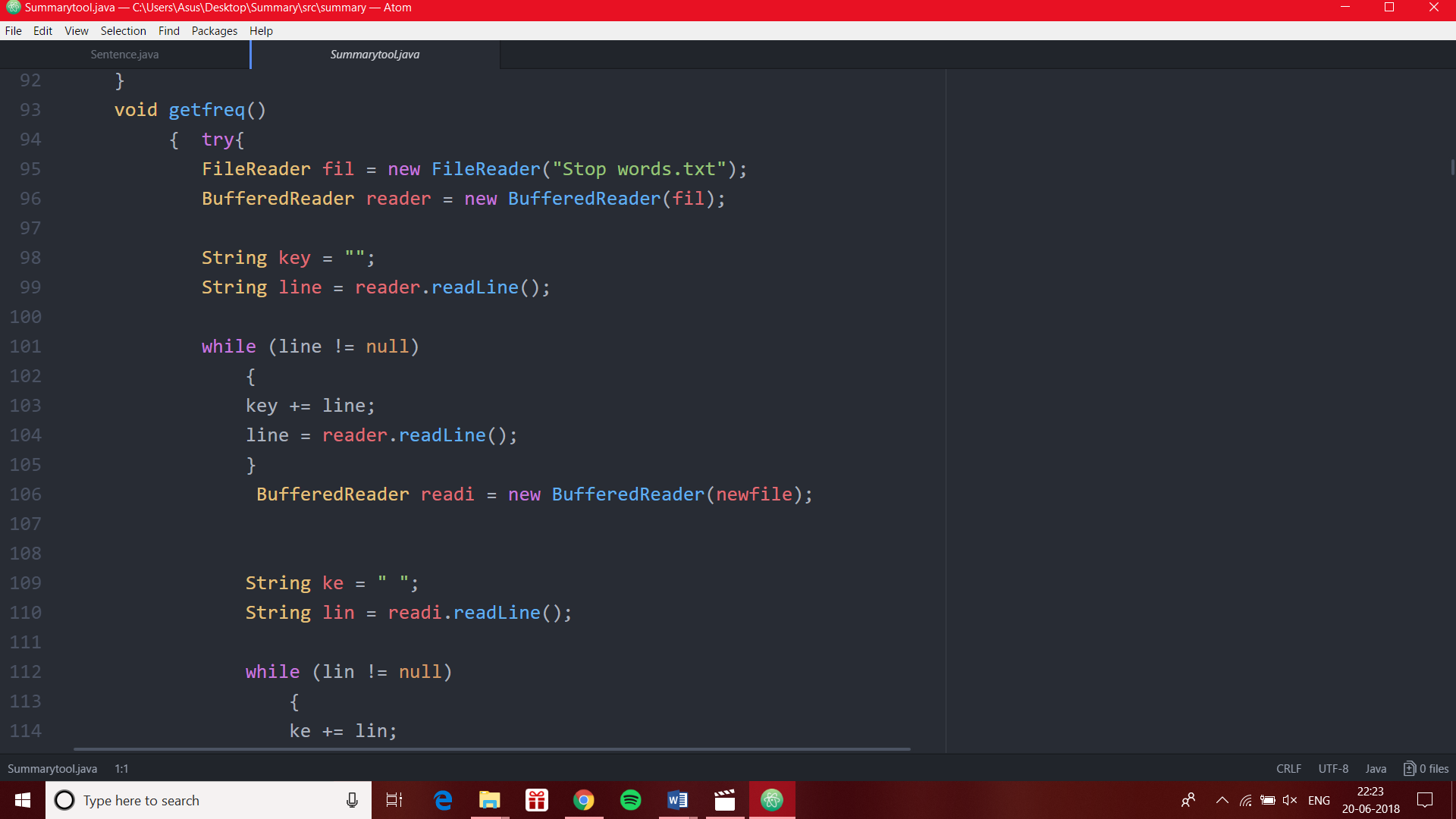


Fig.6: getfreq ( ) method

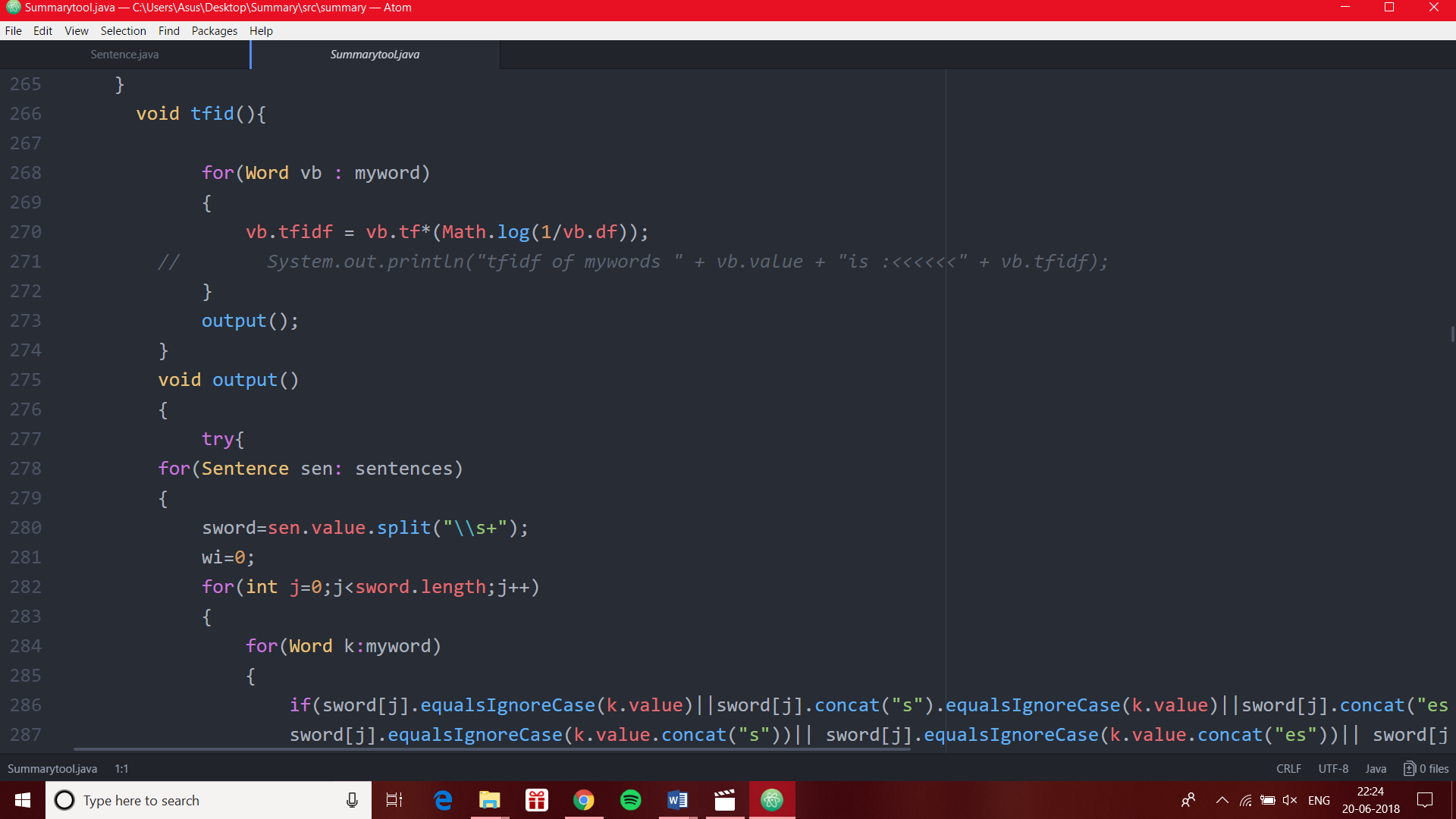


Fig.7: tfidf( ) method and output () method

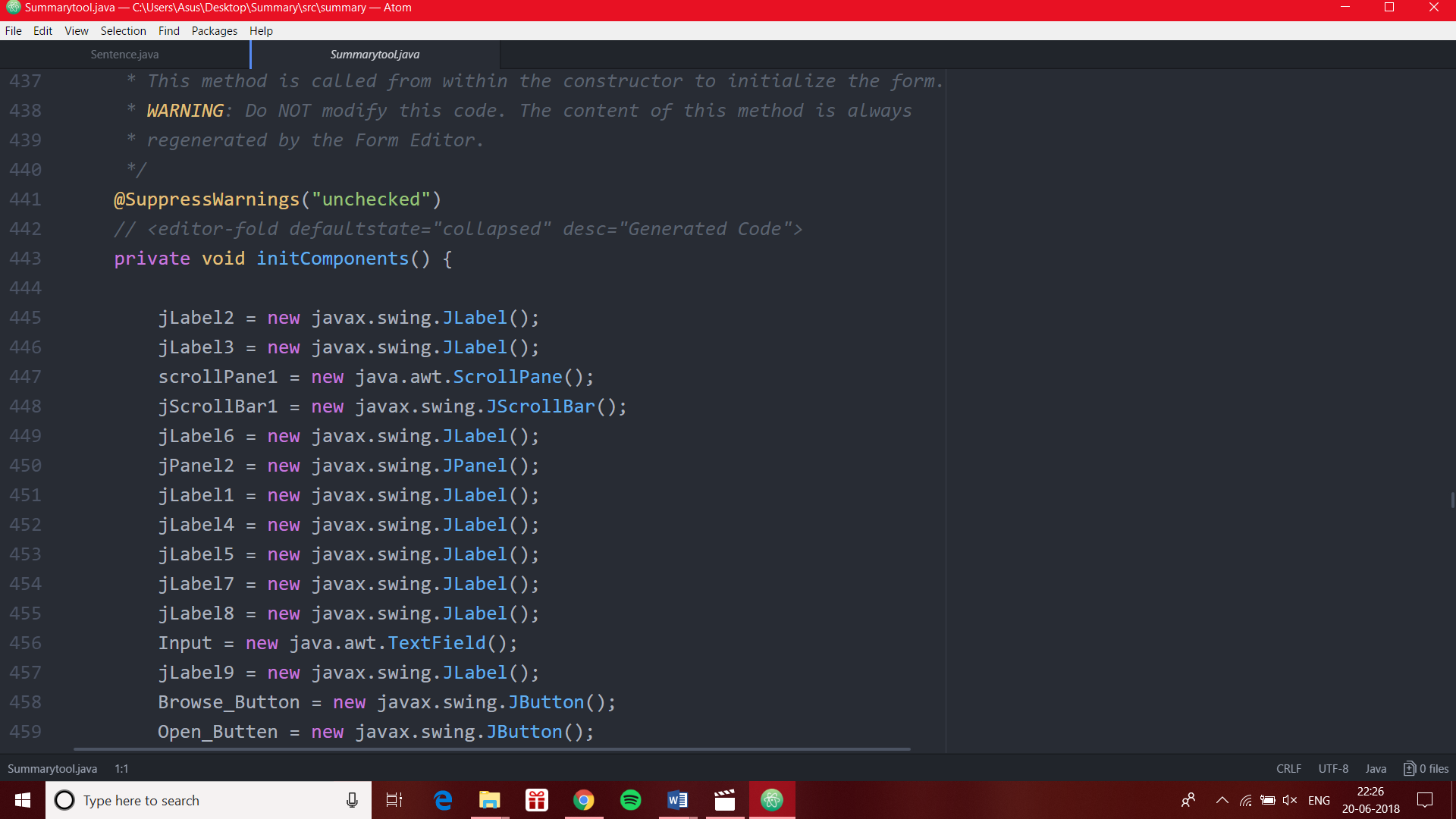


Fig.8: init Components( ) method

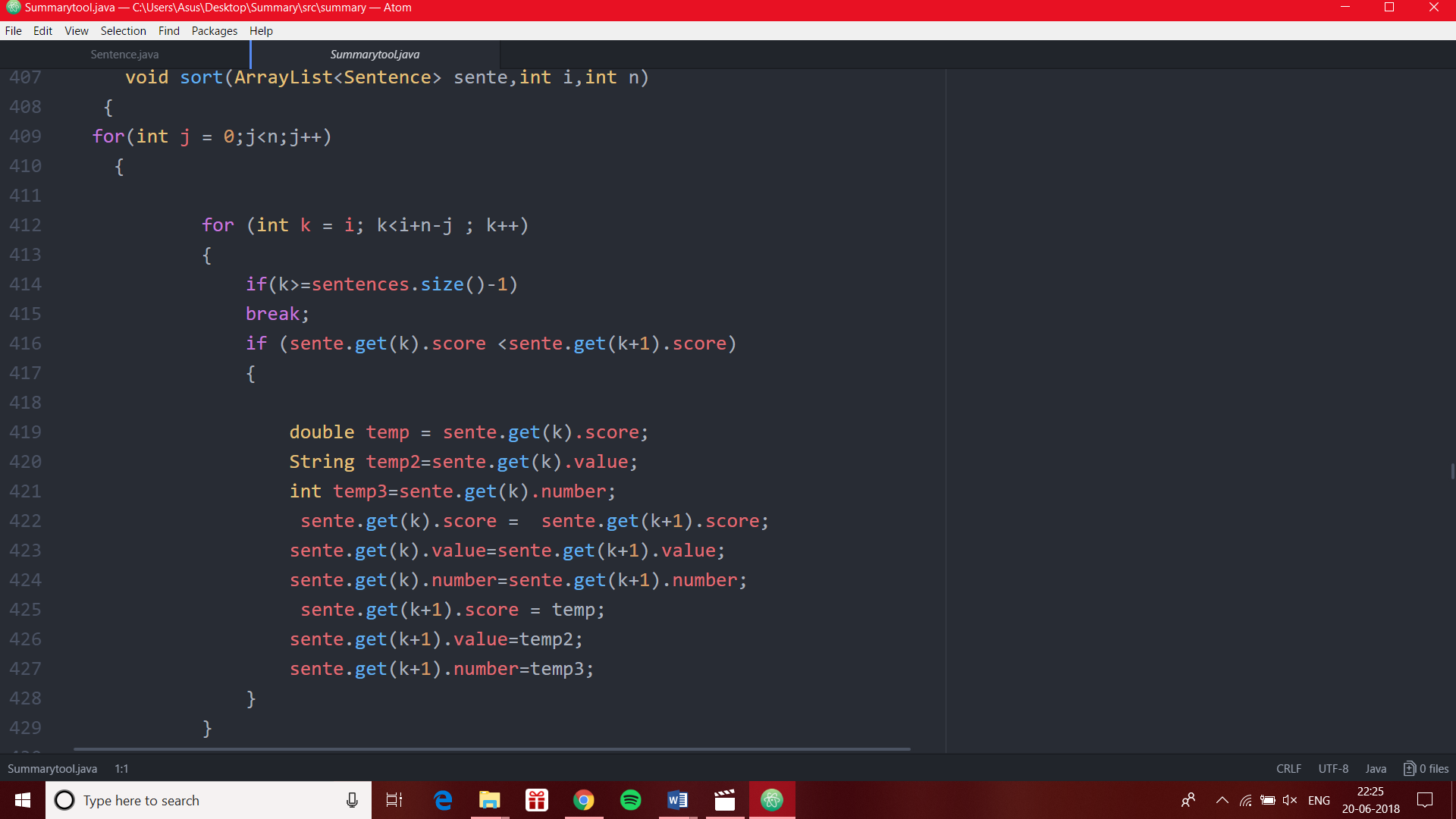


Fig.9: sort( ) method

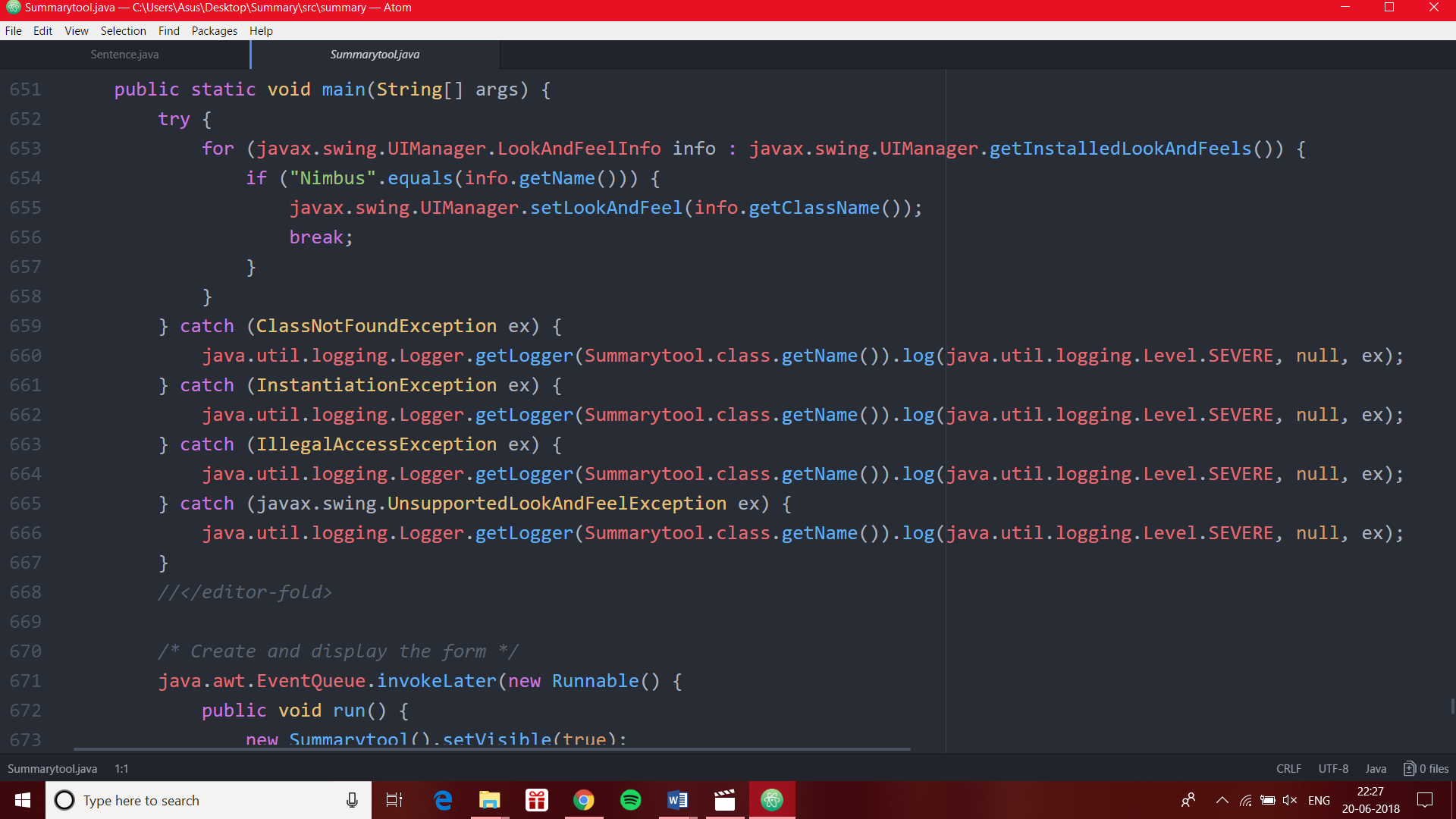


Fig.10: main( ) method

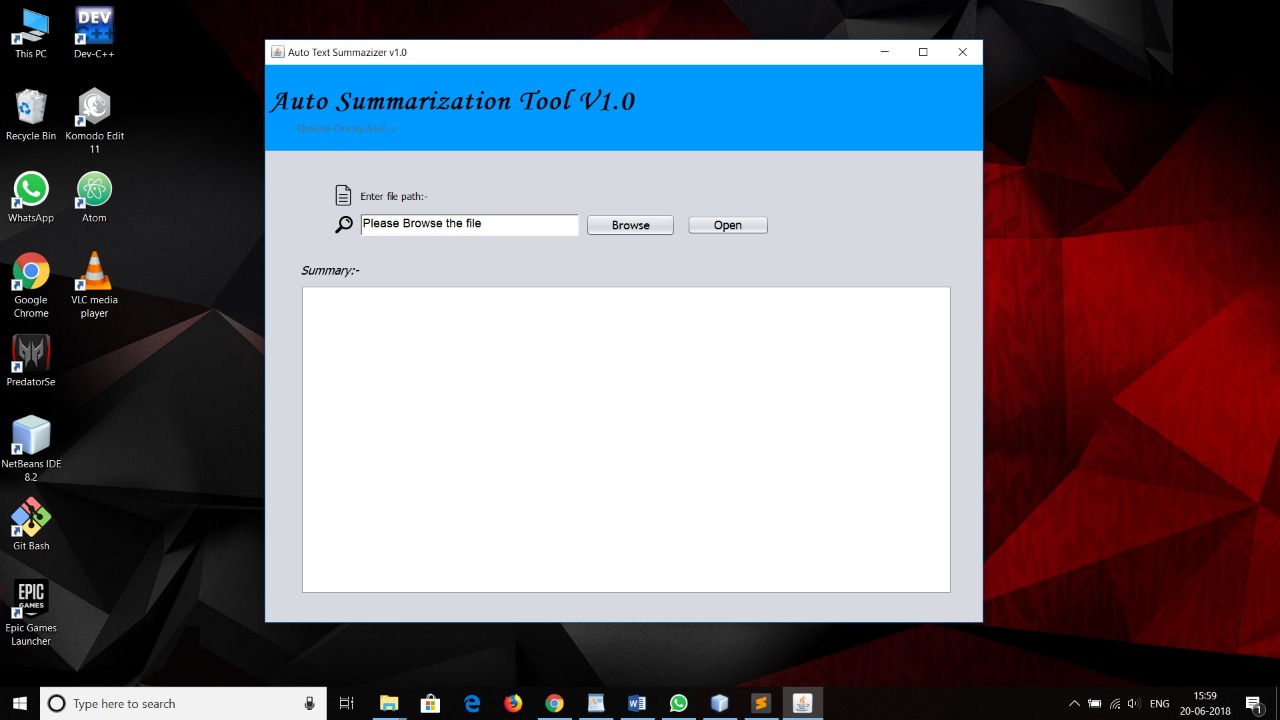


Fig.11: Output Window

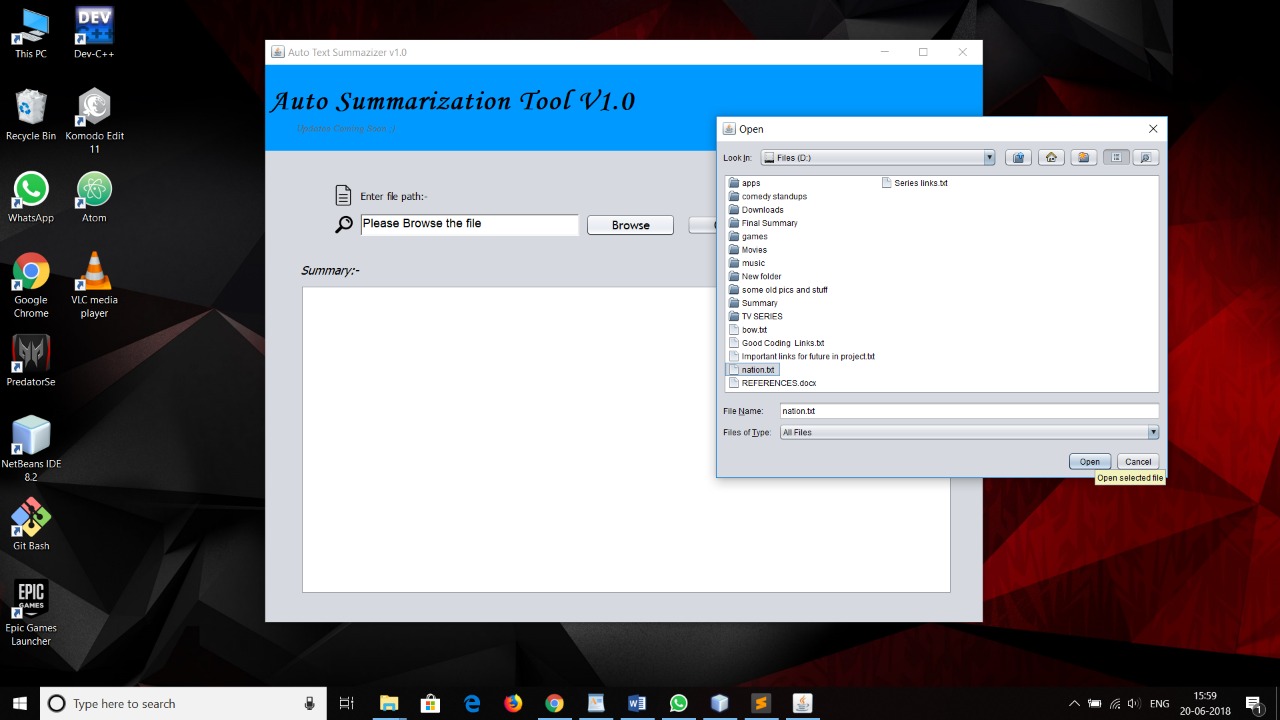


Fig.12: Browse txt file

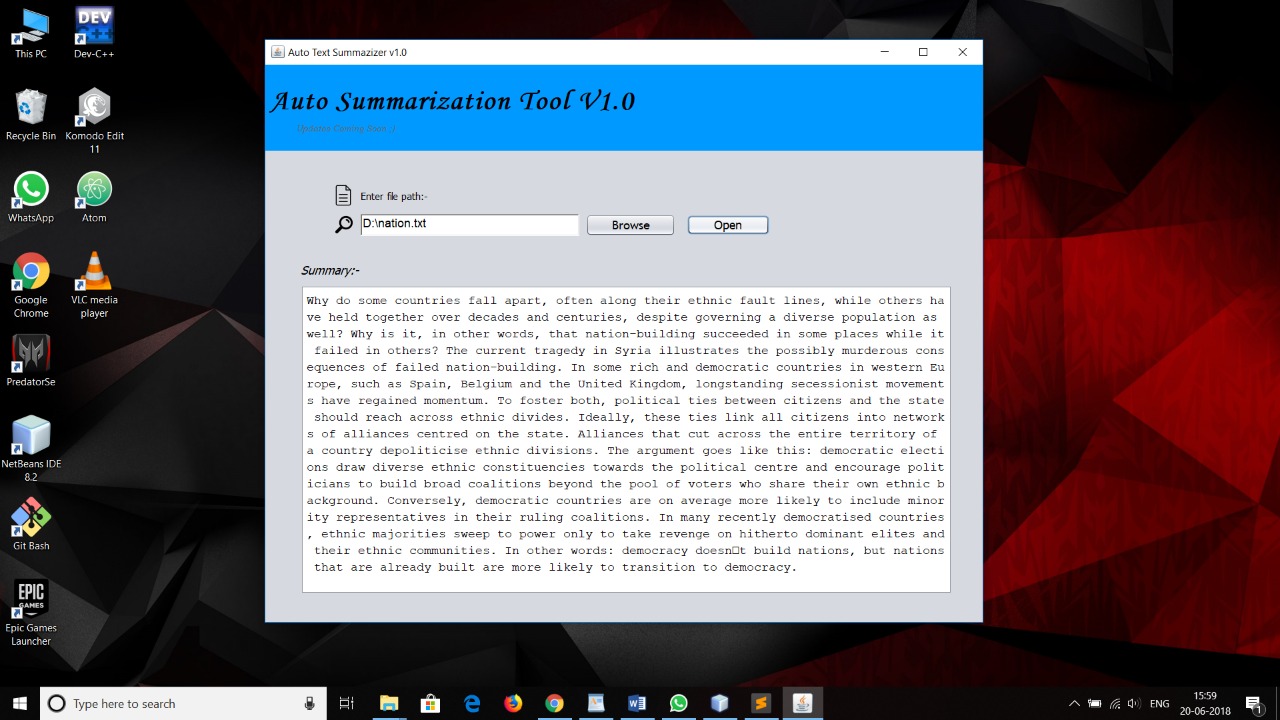


Fig.13: Text Summary in the output window

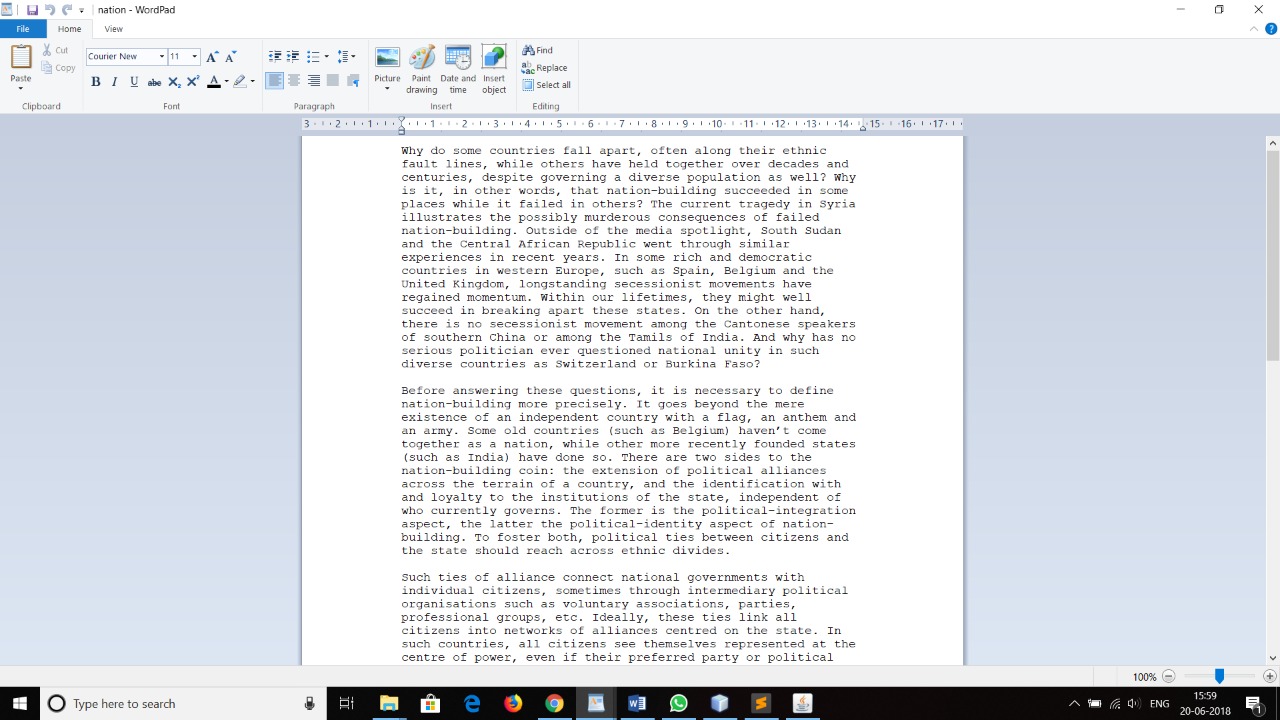


Fig.14: Text Summary as a txt file

1. FUTURE SCOPE

* One of the plans for future may be to apply the topic-focused summarization framework to news articles or blogs and to extend the work in the machine leaning approaches.
* Topic focused summaries of news articles would be lot more accurate and valuable to users. It would be more interesting to work on topic modelling and summarization in the domain of social media in future.
* The state of the art summarization systems is all extractive in nature, but the community is gradually progressing towards abstractive summarization. Although a complete abstractive summarization would require deeper natural language understanding and processing, a hybrid or shallow abstractive summary can be achieved through this page rank method.
* In future work abstractive methods can be implemented. In abstractive method we can build an internal semantic representation and then use natural language generation techniques to create a summary.
* Research in summarization can continue to enhance the diversity and information richness and strive to produce coherent and focused answers to users’ information need.

1. CONCLUSION

The system has developed an automatic text summarization system using page ranking method implemented in java.

This framework is automated to work with the articles of any domain with change in the list of thematic words. It works on the files stored on the secondary memory.

It is done by first cleaning the text file by removing full stop, common words (conjunction, verb, adverb, preposition etc.). Then calculating the frequency of each words and selecting top words which have maximum frequency. Then these frequencies are used to score the sentences. This technique retrieves important sentence emphasizing on high information richness in the sentence as well as high Information retrieval. These related maximum sentences generated scores are clustered to generate the summary of the document.

In frequency-based technique obtained summary makes more

meaning.

The keyword frequency-based summary generation algorithm has been

found to be very simple.

It worked efficiently with larger documents also.

1. REFERENCES
2. For understanding what is TFIDF and how to use it in our Project:
3. <https://www.youtube.com/watch?v=RPMYV-eb6lI>
4. <https://www.youtube.com/watch?v=hXNbFNCgPfY>
5. For understanding what is WordCounter and how to use it in our Project:
6. <https://www.youtube.com/watch?v=fxQ0B6BkfKo>
7. <https://www.youtube.com/watch?v=unm0BLor8aE>
8. For Basic approach

<https://www.youtube.com/watch?v=1PXGcUA3m18>

1. Links of different documents used:
2. <https://en.wikipedia.org/wiki/Java>
3. <https://en.wikipedia.org/wiki/Fortnite>
4. <https://en.wikipedia.org/wiki/Environment>
5. <https://en.wikipedia.org/wiki/Automatic_summarization>
6. <https://en.wikipedia.org/wiki/Engineering>
7. For building User Interface:

<https://netbeans.org/downloads/>

1. Application used:
2. Netbeans(<https://netbeans.org/downloads/>)
3. Bluej (<https://www.bluej.org/download/> )
4. Atom (<https://atom.io/>)
5. Notepad
6. Microsoft Word