Article Generator

Kazi Abir Adnan (安奈威)
2014280162
Machine Learning Project Report
Department of Computer Science & Technology, Tsinghua University

ABSTRACT

This project describes how to join paragraphs together in longer articles. Independent Paragraphs are described separately. In many cases, you cannot fully develop a topic in a single paragraph. Instead, you will find it more useful to think in terms of groups or clusters of paragraphs within your essay.

A paragraph cluster is a group of paragraphs that develop the subtopics of one main idea. There are no rules for the number of paragraphs that an essay must contain. To decide whether to divide one main idea into a paragraph cluster, use the criteria for paragraph unity and coherence.

Article generator is a standalone software to generate article automatically on a given topic. User only gives the topic name to the system and gets an output as an article from the software. It is basically a JAVA application that uses machine learning algorithms to generate the article. The process is unsupervised and do not need any data to train the model. KMeans clustering, Group Average Agglomerative clustering and word2Vector has been used sequentially to generate the article.

Key words: *Article generator, Paragraph Clustering, Sentence to Vector*

1. INTRODUCTION

Article generator is a standalone software to generate article automatically on a given topic. User only gives the topic name to the system and gets an output as an article from the software. It is basically a JAVA application that uses machine learning algorithms to generate the article. The process is unsupervised and do not need any data to train the model. KMeans clustering, Group Average Agglomerative clustering and word2Vector has been used sequentially to generate the article.

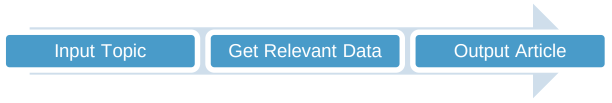
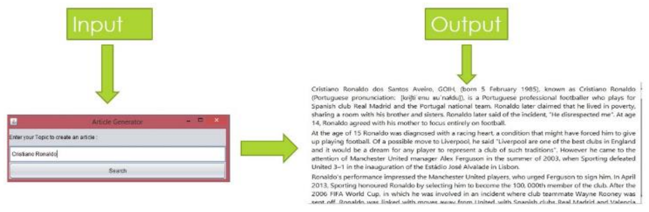


FIGURE 1 : PROCESS

If you want to create an article about great football player 'Cristiano Ronaldo' the process is

- Give input to the system
- Wait for the result
- The system will show you the output in a separate view



Enter your Topic to create an article :
Cristiano Ronaldo

Cristiano Ronaldo dos Santos Aveiro, GOIH, (born 5 February 1985), known as Cristiano Ronaldo (Portuguese pronunciation: [kriʃti'enu ʁu'naɫdu]), is a Portuguese professional footballer who plays for Spanish club Real Madrid and the Portugal national team. Ronaldo later claimed that he lived in poverty, sharing a room with his brother and sisters. Ronaldo later said of the incident, "He disrespected me". At age 14, Ronaldo agreed with his mother to focus entirely on football.

At the age of 15 Ronaldo was diagnosed with a racing heart, a condition that might have forced him to give up playing football. Of a possible move to Liverpool, he said "Liverpool are one of the best clubs in England and it would be a dream for any player to represent a club of such traditions". However he came to the attention of Manchester United manager Alex Ferguson in the summer of 2003, when Sporting defeated United 3–1 in the inauguration of the Estádio José Alvalade in Lisbon.

Ronaldo's performance impressed the Manchester United players, who urged Ferguson to sign him. In April 2013, Sporting honoured Ronaldo by selecting him to become the 100, 000th member of the club. After the 2006 FIFA World Cup, in which he was involved in an incident where club teammate Wayne Rooney was sent off, Ronaldo was linked with moves away from United, with Spanish clubs Real Madrid and Valencia

FIGURE 2 : EXAMPLE

Existing process to create an article is very cumbersome. There are many dependencies which cannot be ignored. Currently what is done is mentioned below:

- Open an editor
- Write something on our own
- Search on internet and get relevant document
- Use relevant information from the document
- Add references of that information
- Rearrange or create paragraphs on subtopics

Now, we want to automate this process. So, user has to just give the input and the output will be the article.

2. HIGH LEVEL DESIGN

The process is a combination of different sub processes. One can think of it as different stages. None of the stages can be ignored or skipped. There are dependencies in the consequent stages and the result of the previous stage is the input of next stage.

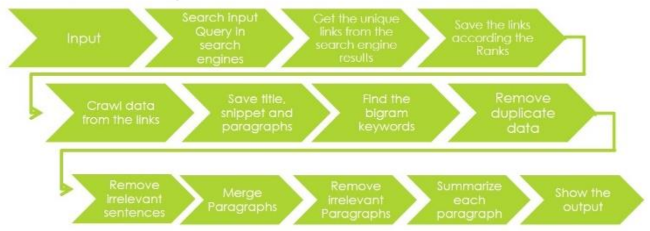

Input
Search Input Query in search engines
Get the unique links from the search engine results
Save the links according the Ranks
Crawl data from the links
Save title, snippet and paragraphs
Find the bigram keywords
Remove duplicate data
Remove irrelevant sentences
Merge Paragraphs
Remove irrelevant Paragraphs
Summarize each paragraph
Show the output

FIGURE 3 : HIGH LEVEL DESIGN

3. DATASET

Data is being crawled from the links of search engine results to get the relevant data for a given topic. Dataset is dynamic and can vary depending on the search results. So, dataset is the search result links of a query topic. 3 search Engines is used to find the links relevant with query topic (Google, Bing and Yahoo). Along with the data, title, snippet and the link addresses of each search engine results is also crawled. Then paragraphs of each saved links is crawled consequently and saved in a list. Moreover, highest ranked documents is emphasized to generate the article from given data.

4. METHODS

This system has 6 main sub process. Each sub process is dependent on its previous process. The steps are mentioned below

- **Preprocessing step** : Removing duplicate paragraphs
- **Keyword Finding** : Finding the keywords from whole corpus
- **Sentence Matching** : Making new paragraph by sentence clustering
- **Remove irrelevant and duplicate sentences** : Removing paragraphs that has irrelevant sentences
- **Merge Paragraphs** : Merging the similar paragraphs from the previous step
- **Summarization** : Summarizing each of the paragraphs remaining important lines

Details of each of these sub processes is mentioned one by one below.

4.1 PREPROCESSING STEP

This is the initial step after crawling the data. This step is mainly to remove the duplicate paragraphs from the input data. Cosine Similarity measure is used to find similarities between raw input paragraphs from different links. A threshold of 85% is chosen to identify the similarity. That means if cosine similarity is more than 85% then data is duplicate. So, the paragraphs which have more than 85% similarity is removed to process it further for the next step.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

FIGURE 4 : COSINE SIMILARITY

Frequency of words is the input vector to measure the cosine similarity. It is important to mention that each of the words has been preprocessed before word counting and analyzing. First edit distance was tried to find similarity. But the result was very poor. Because the length of the paragraph varies a lot and similarity score for this was always low. Any other similarity can be used here rather than Cosine Similarity. Result won't change exceptionally here.

4.2 KEYWORD FINDING

Keywords are necessary to summarize and to find the important lines of the document. Here, frequency of words is the key to find the keywords. To find it each of the words has been preprocessed from every sentences. The preprocess part includes converting each word to lowercase, removing illegal characters, Stemming (KStem), Stop words filtering etc. Bigram keywords is used here to find the keywords from whole corpus. Every combination of two words has been matched to find bigram Keywords. We can try it for ngram keywords. But, it will make the process very slow for nothing. Moreover, for stemming, Porter Stemmer algorithm is not used as it is very hard stemmer to analyze words. That is why we have used KStem algorithm which is very soft stemmer and very good to analyze words. We have also used Apache Lucene token analyzers. Apache Lucene project has excellent Natural language processing library. Lastly it is important to mention that top 20 frequent words are used as keywords in the system. There are many methods to find the keywords from the whole corpus. You can choose any other method to find it. But, the results are more or less similar and it doesn't have a great impact on the final result.

4.3 SENTENCE MATCHING

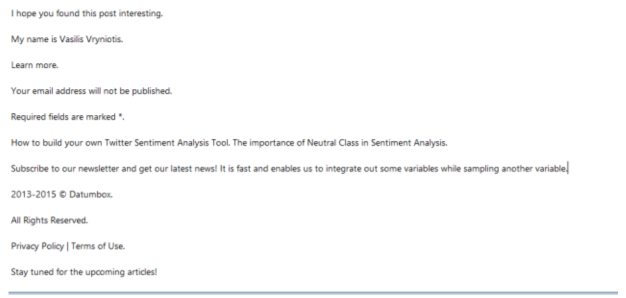
This is a temporary step. It is mainly used to identify the irrelevant and unimportant lines from the input paragraphs. New temporary paragraphs are created using sentence clustering. This step is an Intermediate to remove some of the lines from input paragraphs. Why Sentence Clustering is used? Basically, temporary target is to find similar sentences. So that, we can find the irrelevant and duplicate ones from paragraphs. Now in this step the algorithm will make a cluster of irrelevant sentences (ex. Footer words, web links, email addresses, thank you words or common blog sentences etc.). So, we can easily remove this irrelevant sentences from final article.

Group Average Agglomerative Clustering (GAAC) has been used to cluster sentences.

$$\text{SIM-GA}(\omega_i, \omega_j) = \frac{1}{(N_i + N_j)(N_i + N_j - 1)} \sum_{d_m \in \omega_i \cup \omega_j} \sum_{d_n \in \omega_i \cup \omega_j, d_n \neq d_m} \vec{d}_m \cdot \vec{d}_n$$

FIGURE 5 : GAAC SIMILARITY MATCHING FORMULA

It is a hard clustering and starts with each sentence as a cluster. All pair similarities between sentences are measured from the above formula. Core idea is objects being more related to nearby objects. Merge with each other at certain stages. (Merge until similarity reaches threshold). The threshold here was used is 0.50. It performs well to make cluster of similar sentences. The unnecessary lines can be found very easily.



I hope you found this post interesting.

My name is Vasilis Vryniotis.

Learn more.

Your email address will not be published.

Required fields are marked *.

How to build your own Twitter Sentiment Analysis Tool. The importance of Neutral Class in Sentiment Analysis.

Subscribe to our newsletter and get our latest news! It is fast and enables us to integrate out some variables while sampling another variable.

2013-2015 © Datumbox.

All Rights Reserved.

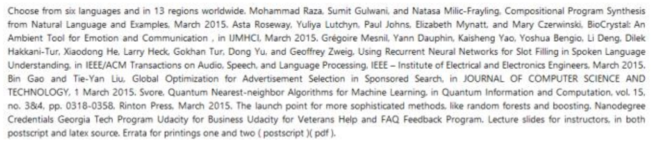
Privacy Policy | Terms of Use.

Stay tuned for the upcoming articles!

FIGURE 6 : RESULT OF THE STEP

4.4 REMOVE IRRELEVANT & DUPLICATE SENTENCES

This system has 6 main sub process. Each sub process is dependent on its previous process. The steps are mentioned below Now, we have some clusters of similar sentences. So, we have to remove the clusters which doesn't have any keywords. That means, a cluster which doesn't contain any relevant information to our topic.



Choose from six languages and in 13 regions worldwide. Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling, Compositional Program Synthesis from Natural Language and Examples, March 2015. Asta Roseway, Yuliya Lutchyn, Paul Johns, Elizabeth Mynatt, and Mary Czerwinski, BioCrystal: An Ambient Tool for Emotion and Communication , in IJMHCI, March 2015. Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig, Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding, in IEEE/ACM Transactions on Audio, Speech, and Language Processing, IEEE – Institute of Electrical and Electronics Engineers, March 2015. Bin Gao and Tie-Yan Liu, Global Optimization for Advertisement Selection in Sponsored Search, in JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, 1 March 2015. Svore, Quantum Nearest-neighbor Algorithms for Machine Learning, in Quantum Information and Computation, vol. 15, no. 3&4, pp. 0318-0358, Rinton Press, March 2015. The launch point for more sophisticated methods, like random forests and boosting. Nanodegree Credentials Georgia Tech Program Udacity for Business Udacity for Veterans Help and FAQ Feedback Program. Lecture slides for instructors, in both postscript and latex source. Errata for printings one and two (postscript)(pdf).

FIGURE 7 : REMOVE IRRELEVANT PARAGRAPHS

We have used a threshold here to find irrelevant clusters. Like, if (0-3) keywords are present in the cluster, then remove it.

4.5 MERGE PARAGRAPHS

Now the next step is to merge the processed paragraphs. "K Means" algorithm is used here to merge the paragraphs. Because KMeans is a centroid based clustering with fixed K. Here, the value of K is fixed to 20. This value means the number of paragraph in the output article. TF-IDF of words has been used as features of KMeans.

The main goal is to just merge the similar paragraphs. Here input data is the varied number of paragraphs after processed by GAAC. Similarity between paragraphs has been measured using Cosine distance of each vectors of the paragraph.

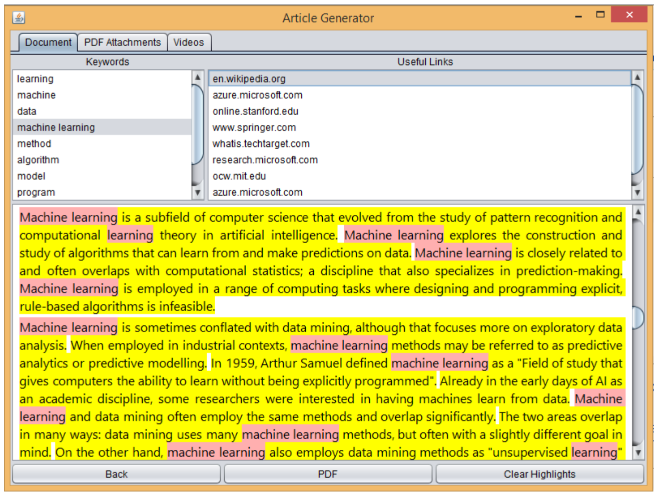
It is important to mention that, here K Means clustering is customized. Each cluster is sorted by the paragraph location and link rank. So, sorting is done for each paragraphs of every clusters by ranking of them. Highest ranked document's information is highly preferred in the output article. Because, these links contain important and most relevant data to given topic.

4.6 SUMMARIZATION

The final and important step is the summarization part. Because, it will make the paragraphs more readable and attractive. So, our goal is to summarize each paragraph to a meaningful one. There are many kinds of automatic summarization processes. We have used here Extract-based summarization. The main technique is to find the significant sentences in each of the paragraphs. Score has been given to each sentences. Scoring is done by keywords of whole corpus and significant words in each paragraph. Significant word is highest frequency words of the paragraph. And keywords are from the whole corpus. If sentence contains keyword, score is added by highest point. Stemming and all kinds of filtering is done here also during score analyzing words and calculation. We have used a compression ratio (like 40%) to calculate the length of summary. This ratio is important to know the length of the summary. Then we sort the scores of each sentences and choose the highest ones in the paragraph. It should be clear that we do not reorder the sentences in this step. We are just picking the important lines for the output paragraph.

5. RESULT

Here is a result of a topic called "Machine Learning"



learning
machine
data
machine learning
method
algorithm
model
program

en.wikipedia.org
azure.microsoft.com
online.stanford.edu
www.springer.com
whatis.techtarget.com
research.microsoft.com
ocw.mit.edu
azure.microsoft.com

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the construction and study of algorithms that can learn from and make predictions on data. Machine learning is closely related to and often overlaps with computational statistics; a discipline that also specializes in prediction-making. Machine learning is employed in a range of computing tasks where designing and programming explicit, rule-based algorithms is infeasible.

Machine learning is sometimes conflated with data mining, although that focuses more on exploratory data analysis. When employed in industrial contexts, machine learning methods may be referred to as predictive analytics or predictive modelling. In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed". Already in the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. Machine learning and data mining often employ the same methods and overlap significantly. The two areas overlap in many ways: data mining uses many machine learning methods, but often with a slightly different goal in mind. On the other hand, machine learning also employs data mining methods as "unsupervised learning"

FIGURE 8 : RESULT OF TOPIC "MACHINE LEARNING"

6. APPLICATION & TOOLS

- JAVA Desktop application (JDK 1.8)
- External libraries
    - JSoup – to crawl data
    - Json parsers
    - Text to pdf converter
    - Apache Common Library
    - Apache Lucene – NLP analyzer
    - Word2Vec, Sentence2Vec library

7. SYSTEM FEATURES

- Highlight keywords
- Highlight sentences by web pages
- Show the original link document in a separate view
- Rearrange the paragraphs manually
- Export as a pdf file
- PDF and video attachments if available
- Spell checking before searching the query topic on search engine

8. COMPARISON WITH EXISTING SYSTEM

There are some online composition generator. Some of these systems just crawls the paragraphs from the link and let users to create an article by choosing from those paragraphs. But, these software's are not free and very hard to find. Moreover, they are not very much user friendly.

9. ANALYSIS

My First try was only Sentence clustering. But, it does not make clusters of sentences together in a meaningful way. Example:

- ***The Dirichlet process can** also be seen as the infinite-dimensional generalization of the Dirichlet distribution. **The Dirichlet process can** be used to circumvent infinite computational requirements as described above. **The Dirichlet Process can** also be used for nonparametric hypothesis testing*

Still, the result of this process is not satisfactory. Because,

- **Transitional words**: Transitional words like accordingly, because of, consequently, thus etc. should be handled as a special case. These words has dependency with previous sentences. So we have to map the dependency in this model.
- **Paragraph flow:** To make a meaningful article we have to maintain the flow of the paragraph. Like Introduction, Description, Conclusion etc. Paragraphs might have dependency with the previous one. So, we have to maintain the dependency.

Moreover, article generation is very inconsistent. Because, whole system is dependent on the result of search engines. So, if it cannot find any good links then the result article will be very poor. The system is Very slow if input paragraph number is very large (like 400 paragraph).

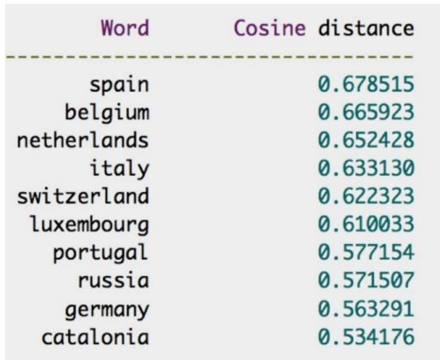
10. IMPROVEMENT

We are using only word frequency and TF-IDF as our feature. So, by changing the features might improve the result. Previously we have used TF-IDF as our feature vector. Probability cannot extract the meaning of the whole paragraph only by this feature. So, to improve the result Sentence2Vec (extend from Word2Vec) is used to generate the feature of the paragraphs. This feature is fed to K-means cluster (feature size of sentence2vec is less than TF-IDF). We believe the generated vector of the sentence is more representative of its meaning.

10.1 WORD2VEC

Unsupervised algorithm for learning the meaning behind words. Efficient implementation of the continuous bag-of-words and skip-gram architectures for computing **vector representations** of words.

The words similar to the word "France" using this features are:



Word	Cosine distance
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

FIGURE 9 : SIMILAR WORDS TO FRANCE

Interesting properties of the word vectors:

- *vector('Paris') - vector('France') + vector('Italy') ~ vector('Rome')*
- *vector('king') - vector('man') + vector('woman') ~ vector('queen')*

It first constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications.

It is skip-gram model. The training objective of skip-gram is to learn word vector representations that are good at predicting its context in the same sentence. Mathematically, given a sequence of training words *w*1, *w2*, …, *wT*, the objective of the skip-gram model is to maximize the average log-likelihood where *k* is the size of the training window.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=-k}^{j=k}\log p(w_{t+j}|w_t)$$

FIGURE 10 : LOG LIKELIHOOD

In the skip-gram model, every word *w* is associated with two vectors u_w and v_w which are vector representations of *w* as word and context respectively. The probability of correctly predicting word w_i given word w_j is determined by the soft max model, which is where *V* is the vocabulary size Softmax Model complexity: log$p(w_i|w_j)$ proportional to *V*, which can be easily in order of millions. Hierarchical softmax, which reduced the complexity to $O(\log(V))$

$$p(w_i|w_j) = \frac{\exp(u_{w_i}^{\top} v_{w_j})}{\sum_{l=1}^{V}\exp(u_l^{\top} v_{w_j})}$$

FIGURE 11 : PROBABILITY

Modify the algorithm of the word2Vec **unsupervised learning of continuous representations for larger blocks of text**, such as sentences, paragraphs or entire documents. Use a vector to represent the sentence. One exception to this rule are the parameters relating to the training method used by the model. In the word2vec architecture, the two algorithm names are **“continuous bag of words”** (cbow) and **“skip-gram”** (sg); in the doc2vec architecture, the corresponding algorithms are **“distributed memory”** (dm) and **“distributed bag of words”** (dbow). Since the distributed memory model performed noticeably better in the paper, that algorithm is the default when running Doc2Vec.

11. FUTURE PERSPECTIVE

This idea has a very good and noble future perspective. There are not many works in this field. There are many works on document clustering, sentence clustering. But, paragraph clustering is a very challenging task and not many people tried it. So, if someone can work in this area then it will be very helpful and successful project for some of the useful applications like:

- Can be used as a wiki page generator for a topic
- Can be used in social search engine to create automatic page(ex. Facebook)
- Can help bloggers to suggest article paragraphs before writing (Like searching suggestion)
- Can be used to solve assignments!

12. CONCLUSION

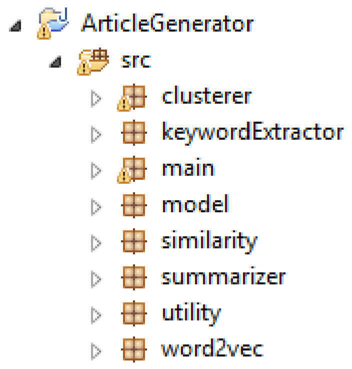
Basically this application is related to natural language processing. It is not an easy task to implement only by unsupervised learning. There should be some training for this model to know how to create an article and the flow of the paragraphs. Moreover, sentence dependency is also an important aspect of an article. Though our system has shortcomings, it can generate a sober article which is very useful. It can help users to gather idea and modify more to create a more attractive article. There are lots of scope to improve the performance of this system. I think it is a great opportunity to contribute in the area of Paragraph Clustering.

13. REFERENCES

1. Quoc Le, Tomas Mikolov, “Distributed Representations of Sentences and Documents”
2. Web document clustering: a feasibility demonstration
3. Stanford Group average agglomerative clustering

14. CODE EXPLAINATION

The code base has different packages. Every package has different functionality. I tried to make the project independent of each other. So that we can apply any method to create the algorithm. So, code structure is very portable and you can add or remove functionalities very easily.

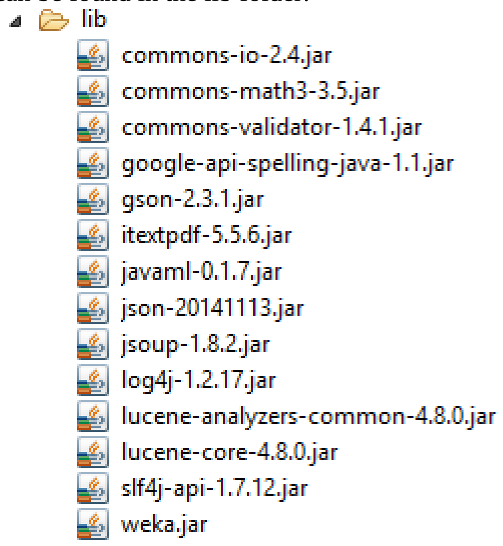


```
ArticleGenerator
  src
    clusterer
    keywordExtractor
    main
    model
    similarity
    summarizer
    utility
    word2vec
```

FIGURE 12 : PACKAGES OF SOURCE CODE

- Clusterer: Contains cluster algorithms. Different processes has been tried and the algorithms can be found here. It includes sentence and paragraph clustering algorithms.
- KeywordExtractor : Finds the keyword from whole corpus
- Main : Contains the main file, article generator algorithm and the GUI files
- Model : contains the data structure of objects
- Similarity : Contains the code to find the similarity between paragraphs and also for sentences
- Summarizer : This package contains the code to summarize a paragraph
- Utility : Contains some useful functions which has been used in different packages
- Word2vec : This is used to find the features of each word from sentences

Many Java libraries have been used in the project to make it simpler to understand. Library that has been used in the project can be found in the lib folder.



```
lib
    commons-io-2.4.jar
    commons-math3-3.5.jar
    commons-validator-1.4.1.jar
    google-api-spelling-java-1.1.jar
    gson-2.3.1.jar
    itextpdf-5.5.6.jar
    javaml-0.1.7.jar
    json-20141113.jar
    jsoup-1.8.2.jar
    log4j-1.2.17.jar
    lucene-analyzers-common-4.8.0.jar
    lucene-core-4.8.0.jar
    slf4j-api-1.7.12.jar
    weka.jar
```

FIGURE 13 : LIBRARIES USED IN THE PROJECT

Some of the main class’s description is given below to help you to understand the model.

14.1 MAIN FUNCTION

Some of the main class's description is given below to help you to understand the model. To start the code you have to run the Application.java from main package.

```
public class Application {

	public static void main(String[] args) {

		try {
			for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
				if ("Nimbus".equals(info.getName())) {
					UIManager.setLookAndFeel(info.getClassName());
					break;
				}
			}
		} catch (Exception e) {
		}

		InputUI entry = new InputUI();
		entry.setVisible(true);
	}
}
```

FIGURE 14 : MAIN FUNCTION

14.2 SEARCH ENGINE API KEY

Some of the main class's description is given below to help you to understand the model. You have to provide account key for Bing to get the result from Bing search engine. You will find it in model package → BingSearch.java class

```
String accountKey = "43hZISppxtGLFaLc4uVUH5TZ24i3tJ+tL8KlwptMAdo";
		String bingUrlPattern =
"https://api.datamarket.azure.com/Bing/Search/Web?Query=%%27%s%%27&$format=JS
ON";
```

FIGURE 15 : API KEY

14.3 CRAWLING SEARCH ENGINE

Some of the main class's description is given below to help you to understand the model. User agent is changed to crawl the result from search engines page. Otherwise search engines won't allow you to crawl data. Because it is not legal. There is a great scope to improve crawling technique. More data can be grabbed improving the data crawler. Again, multithreading can be used to improve the running time. I didn't use any threading for simplicity.

```
String request = "https://www.google.com/search?q=" + queryString
				+ "&num=10";
		try {
			Document doc = Jsoup
					.connect(request)
					.userAgent(
							"Mozilla/5.0 (compatible; Googlebot/2.1;
+http://www.google.com/bot.html)")
					.timeout(5000).get();

			Elements divs = doc.select("li.g");
		} catch (IOException e) {
			e.printStackTrace();
		}
```

FIGURE 16 : SEARCH ENGINE CRAWLING CODE

14.4 WEB LINK OBJECT

Link class contains the crawled data of each link. It contains all the data crawled from a single link. I save each of the links according to the page ranks given by the search engines. It will help to cluster the paragraphs in later stage. You will find it in model package

```
public class Link {
	private int id;
	private String url;
	private String title;
	private String snippet;
	private String description;
	private String searchEngine;
	private int rank;
	private ArrayList<String> paragraphs = new ArrayList<String>();
	private String contentType = null;

	public Link(String searchEngine, String url, String title, String snippet,
			int rank) {
		this.url = url;
		this.title = title;
		this.snippet = snippet;
		this.searchEngine = searchEngine;
		this.rank = rank;
	}

	public String getURL() {
		return this.url;
	}

	public String getTitle() {
		return this.title;
	}

	public String getSnippet() {
		return this.snippet;
	}

	public String getSearchEngine() {
		return this.searchEngine;
	}

	public void setDescription(String description) {
		this.description = description.replace("<p>", System.lineSeparator())
				.replace("</p>", System.lineSeparator());
	}

	public String getDescription() {
		return this.description;
	}

	public int getRank() {
		return this.rank;
	}

	public void addParagraph(String paragraph) {
		this.paragraphs.add(paragraph);
	}

	public ArrayList<String> getParagraphs() {
		return this.paragraphs;
	}

	public void addContentType(String content) {
		this.contentType = content;
	}

	public String getContentType() {
		if (this.contentType == null) {
			return "Not available";
		} else
			return this.contentType;
	}

	public void setId(int id) {
		this.id = id;
	}
```

FIGURE 17 : WEB LINK CLASS

14.5 ARTICLE CLASS

I have a class named Article.java which is the output of any stages in the sub process and which is showed in the result UI. You can find it in the model package.

```
public class Article {
    private ArrayList<Paragraph> paragraphs;

    public <T> Article(ArrayList<T> list) {
        Object obj = null;
        for (Object o : list) {
            obj = o;
            break;
        }
        if (obj instanceof model.Paragraph) {
            this.paragraphs = (ArrayList<Paragraph>) list;
        } else if (obj instanceof model.Link) {
            int lineNo = 0;
            paragraphs = new ArrayList<Paragraph>();
            ArrayList<Link> webLinks = (ArrayList<Link>) list;
            for (int i = 0; i < webLinks.size(); i++) {
                Link link = webLinks.get(i);
                for (int j = 0; j < link.getParagraphs().size(); j++) {
                    String linkParagraph = link.getParagraphs().get(j);
                    Paragraph articleParagraph = new Paragraph(link.getRank()
                            + j);
                    String text = linkParagraph;
                    BreakIterator iterator = BreakIterator
                            .getSentenceInstance(Locale.US);
                    String source = text;
                    iterator.setText(source);
                    int start = iterator.first();
                    for (int end = iterator.next(); end != BreakIterator.DONE;
start = end, end = iterator
                            .next()) {
                        String line = source.substring(start, end);
                        if (line.length() > 4) {
                            articleParagraph.addSentence(new
Sentence(link
                                    .getId(), new
StringBuilder(line), link
                                    .getRank(), lineNo++));
                        }
                    }
                    articleParagraph.setRank(link.getRank());
                    articleParagraph.setLinkId(link.getId());
                    if (paragraphs.size() < 200) {
                        if (articleParagraph.getSentences().size() > 0) {
                            paragraphs.add(articleParagraph);
                        }
                    }
                }
            }
        }
    }

    public Article() {
        paragraphs = new ArrayList<Paragraph>();
    }

    public ArrayList<Paragraph> getParagraphs() {
        return this.paragraphs;
    }

    public void addParagraph(Paragraph p) {
        this.paragraphs.add(p);
    }
}
```

FIGURE 18 : ARTICLE CLASS

This article class contains paragraph which is defined in Paragraph class and Paragraph class contains list of Sentence object. Basically Sentence object is the sentences crawled from input paragraph.

14.6 ARTICLE GENERATOR FUNCTION

In main package → ArticleGenerator.java file you will find the main generate function to create the article. You can fine tune the code here. Like which method you want to use to create the article or if you want to remove or skip any step.

```
public void generate() {
        //crawls the data
        crawler = new Crawler(query, task, monitor);
        crawler.startWork();

        //Find the keywords
        task.set(40);
        generateKeywords();
        article = new Article(crawler.getResults());

        //Use GAAC to find unnecessary lines
        task.set(50);
        removeUnncessaryLines();
        task.set(80);

        // K means Cluster method with sentence2vec
        // KmeansClusterProcessor kcp = new
KmeansClusterProcessor(article);
        // article = kcp.run(true, 20);
        // Article test2 = kcp.run(false, 20);

        // K Means with TF-IDF
        ClusterDocuments cluster = new ClusterDocuments(article);
        ClusterList clusterlist = cluster.performCluster();
        article = clusterlist.getArticle();
        task.set(90);

        // Summarize paragraphs
        article = SummarizeDocuments.summarize(article, keywords);

        // Remove irrelevant paragraphs
        article = RemoveIrrelevancy.removeIrrelevantParagraphs(article,
                keywords);

    }
```

FIGURE 19 : CODE TO GENERATE THE ARTICLE

A jar has is attached so that you can run the program easily. But you have to make sure that you have installed java in your system.