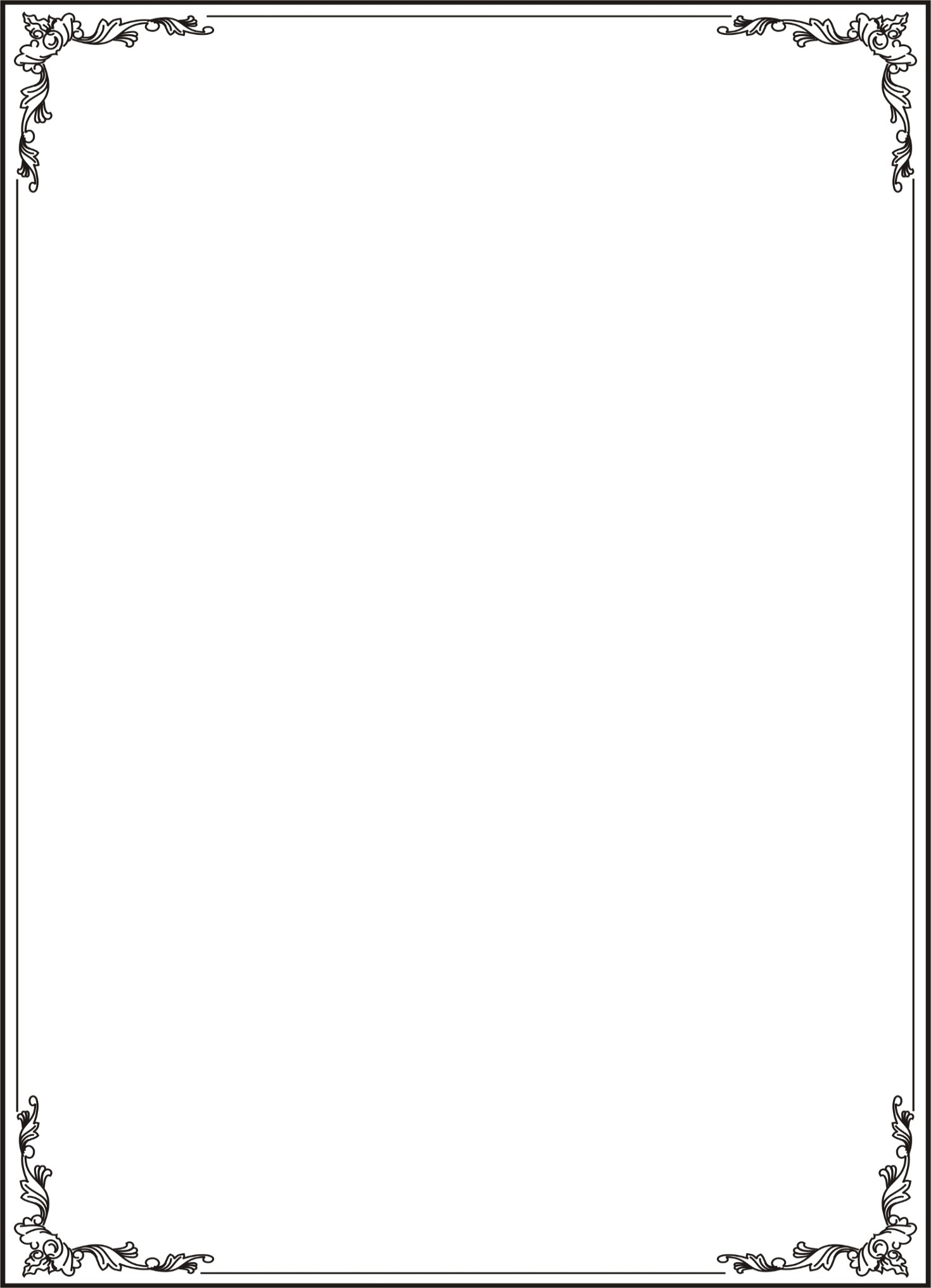
**TRƯỜNG ĐẠI HỌC ĐIỆN LỰC**

Khoa Công Nghệ Thông Tin

-----🙞🙜🕮🙞🙜-----

**BÁO CÁO MÔN MÁY HỌC**

**Đề Tài**

**Chương trình mô phỏng tính toán độ tương tự giữa hai đối tượng**

Giảng viên hướng dẫn : TS. Nguyễn Thị Thu Hà

Sinh viên thực hiện  : Đinh Văn Đông - 1581310014

Trần Thị Diệu Ninh - 1581310041

Lớp : D10CNPM

***Hà Nội, tháng 06 năm 2018***

# LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô trong khoa Công nghệ thông tin, trường Đại học Điện Lực, đặc biệt là cô Nguyễn Thị Thu Hà đã giúp đỡ và hướng dẫn chúng em hoàn thành bài báo cáo môn Máy học.

Thực hiện đồ án là cơ hội để chúng em áp dụng, tổng hợp các kiến thức đã học trên lớp, đồng thời đúc kết được những bài học thực tế phục vụ cho việc học tập và làm việc sau này. Mặc dù đã rất cố gắng để thực hiện đề tài nhưng vì năng lực và thời gian còn hạn chế nên chương trình khó tránh khỏi những sai xót, rất mong thầy (cô) thông cảm. Những góp ý của thầy (cô) là bài học, là hành trang để chúng em vững bước vào cuộc sống sau này. Qua đây chúng em xin trân trọng cảm ơn cô Nguyễn Thị Thu Hà, người đã nhiệt tình hướng dẫn, chỉ bảo chúng em trong quá trình thực hiện đề tài.

Chúng em xin chân thành cảm ơn!

Hà Nội, tháng 6 năm 2018

Sinh viên thực hiện

Đinh Văn Đông

Trần Thị Diệu Ninh

# CHƯƠNG I. TỔNG QUAN MÁY HỌC

## **1.1. Khái niệm máy học**

* Máy học **là một phần của trí tuệ nhân tạo trong đó các thuật toán máy tính được sử dụng để tự học từ dữ liệu và thông tin. Trong machine learning, máy tính không cần phải được lập trình một cách rõ ràng mà có thể tự thay đổi và cải thiện các thuật toán của mình.**
* Ngày nay, các thuật toán machine learning cho phép máy tính giao tiếp với con người, tự lái xe, viết và đăng các bản tin về các trận thi đấu thể thao, hay thậm chí tìm kiếm kẻ tình nghi khủng bố. Chắc chắn rằng học máy sẽ tác động mạnh mẽ lên hầu hết các ngành và các công việc của mỗi ngành. Đó là lý do tại sao mọi nhà quản lý cần phải có hiểu biết tối thiểu về machine learning và sự tiến hoá của nó.
* Học máy có liên quan lớn đến [thống kê](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_Th%E1%BB%91ng_k%C3%AA), vì cả hai lĩnh vực đều nghiên cứu việc phân tích dữ liệu, nhưng khác với thống kê, học máy tập trung vào sự phức tạp của các giải thuật trong việc thực thi tính toán. Nhiều bài toán suy luận được xếp vào loại bài toán NP-khó, vì thế một phần của học máy là nghiên cứu sự phát triển các giải thuật suy luận xấp xỉ mà có thể xử lý được
  1. **Các loại thuật toán học máy**
* Học có giám sát: thuật toán tạo ra một hàm ánh xạ dữ liệu vào tới kết quả mong muốn. Một phát biểu chuẩn về một việc học có giám sát là bài toán phân loại: chương trình cần học (cách xấp xỉ biểu hiện của) một hàm ánh xạ một vector tới một vài lớp bằng cách xem xét một số mẫu dữ liệu - kết quả của hàm đó.
* Học không giám sát: mô hình hóa một tập dữ liệu, không có sẵn các ví dụ đã được gắn nhãn.
* Học nửa giám sát: kết hợp các ví dụ có gắn nhãn và không gắn nhãn để sinh một hàm hoặc một bộ phân loại thích hợp.
* Học tăng cường: trong đó, thuật toán học một chính sách hành động tùy theo các quan sát về thế giới. Mỗi hành động đều có tác động tới môi trường, và môi trường cung cấp thông tin phản hồi để hướng dẫn cho thuật toán của quá trình học.
* Chuyển đổi: tương tự học có giám sát nhưng không xây dựng hàm một cách rõ ràng. Thay vì thế, cố gắng đoán kết quả mới dựa vào các dữ liệu huấn luyện, kết quả huấn luyện, và dữ liệu thử nghiệm có sẵn trong quá trình huấn luyện.
* Học cách học: trong đó thuật toán học thiên kiến quy nạp của chính mình, dựa theo các kinh nghiệm đã gặp.

**CHƯƠNG II. CHƯƠNG TRÌNH MÔ PHỎNG TÍNH TOÁN ĐỘ TƯƠNG TỰ GIỮA HAI ĐỐI TƯỢNG**

* 1. **Tổng quan về độ tương tự**

### **2.1.1 Khái niệm độ tương tự**

Độ tương tự là một khái niệm quan trọng và đã được sử dụng rộng rãi. Các định nghĩa trước đây về độ tương tự được trói buộc trong một ứng dụng cụ thể hoặc một dạng thể hiện của tri thức.

Nhiều độ đo độ tương tự đã được đưa ra, chẳng hạn như nội dung thông tin (Resnik, 1995), độ đo thông tin chung (mutual information – Hindle, 1990), độ đo 7 dựa trên khoảng cách (Lee et al., 1998; Rada et al 1998) và mô hình đặc trưng tương phản (Tversky, 1977). McGill etc đã khảo sát và so sánh 67 độ đo độ tương tự đã sử dụng trong tìm kiếm thông tin (McGill et al, 1979).

Một vấn đề trong độ đo độ tương tự trước đây là một trong số chúng bị trói buộc trong một ứng dụng cụ thể hoặc đảm đương một mô hình cụ thể .Ví dụ độ đo về độ tương tự giữa các khái niệm dựa trên khoảng cách (Lee et al,1989; Rada et al, 1989) thừa nhận rằng phạm vi đã được thể hiện trong một mạng. Nếu một tập các tài liệu không được thể hiện như một mạng, độ đo dựa trên khoảng cách sẽ không được áp dụng. Hệ số dice (súc sắc) và hệ số cosin chỉ có thể áp dụng khi các đối tượng được thể hiện như các vector đặc trưng bằng số.

Một vấn đề khác với các độ đo độ tương tự trước đây là các điều giả định cơ bản của chúng thường không ở trạng thái rõ ràng. Ngoài việc biết các giả định này, không thể tạo ra sự tranh luận về mặt lý thuyết hay phản đối bất cứ độ đo cụ thể nào. Hầu hết tất cả các so sánh và đánh giá của các độ đo độ tương tự trước đây đều dựa trên kết quả do kinh nghiệm.

Định nghĩa độ tương tự đạt được hai mục đích:

* Tính phổ biến (Universality): chúng ta định nghĩa độ tương tự trong thuật ngữ lý thuyết thông tin. Điều đó có thể được áp dụng miễn là phạm vi có một mô hình xác suất. Từ đó lý thuyết xác suất có thể được tích hợp với nhiều loại thể hiện của tri thức, chẳng hạn như thứ tự logic (Bacchus, 1988) và mạng ngữ nghĩa (Pearl, 1988), định nghĩa của chúng ta về độ tương tự có thể được áp dụng cho nhiều loại lĩnh vực mà mỗi độ đo có một giả định riêng trước. Hơn nữa, tính phổ biến của định nghĩa còn cho phép độ đo được sử dụng trong lĩnh vực không có giả định trước, chẳng hạn như độ tương tự giữa các giá trị có thứ tự.
* Tính giả định (Theoretical Justification): độ đo độ tương tự không được định nghĩa một cách trực tiếp bởi công thức. Hơn nữa, nó được phân phát từ một tập các giả định về độ tương tự. Mặt khác, nếu các giả định được cho là hợp lý, độ đo độ tương tự cần thiết phải xảy ra.

#### **2.1.1.1 Định nghĩa độ tương tự (Definition of similarity)**

Mục đích của chúng ta là cung cấp định nghĩa chính thức về khái niệm độ tương tự, đầu tiên chúng ta đưa ra các trực giác (intuitions) về độ tương tự.

* Trực giác 1: Độ tương tự giữa A và B có liên quan tới sự tương đồng của chúng. Sự tương đồng càng nhiều, độ tương tự càng lớn.
* Trực giác 2: Độ tương tự giữa A và B có liên quan tới những sự khác biệt giữa chúng. Càng nhiều sự khác biệt, độ tương tự càng thấp.
* Trực giác 3: Độ tương tự lớn nhất giữa A và B đạt được khi A và B giống hệt nhau (đồng nhất - identical)

#### **2.1.1.2 Độ tương tự giữa các giá trị có thứ tự ưu tiên (Ordinal values)**

Rất nhiều các đặc trưng có các giá trị ưu tiên. Ví dụ, thuộc tính “chất lượng” có thể mang một trong các giá trị sau: “excellent”, “good”, “average”, “bad”, “awful”. Không có một định nghĩa nào ở trên cung cấp độ đo độ tương tự giữa hai giá trị có thứ tự. Bây giờ chúng ta sẽ chỉ ra cách định nghĩa của chúng ta có thể được áp dụng.

Nếu “chất lượng của X là excellent” và “chất lượng của Y là average”, sự mô tả cụ thể nhất của cả X và Y là “chất lượng của X và Y ở giữa hai giá trị excellent và average”. Do đó, sự tương đồng giữa hai giá trị ưu tiên được giới hạn bên trong giữa chúng.

#### **2.1.1.3 Độ tương tự chuỗi (String similarity- A case study)**

Xem xét công việc tìm kiếm từ một danh sách từ các từ mà được xuất phát từ cùng một gốc như là một từ cho sẵn.

Ví dụ, cho trước từ “eloquently”, mục đích của chúng ta là để tìm ra các từ liên quan khác như “ineloquent”, “ineloquently”, “eloquent” và “eloquence”. Để làm điều đó, ta có thể định nghĩa độ đo tương tự giữa hai chuỗi và xếp hạng các từ trong danh sách từ theo thứ tự giảm dần của độ tương tự với từ cho sẵn. Những từ xuất phát từ cùng một từ gốc nên xuất hiện sớm trong bảng xếp hạng.

* + 1. **Độ tương tự ngữ nghĩa**

Độ tương tự ngữ nghĩa là một khái niệm ở đó tập các tài liệu hoặc các thuật ngữ trong một danh sách các thuật ngữ được gán một tỷ lệ dựa trên sự giống nhau về nội dung ý nghĩa của chúng.

Độ đo độ tương tự ngữ nghĩa gần đây được áp dụng và phát triển trong rất nhiều lĩnh vực như trong y học (so sánh các gen), trong phân lớp văn bản (các văn bản tương tự nhau thì cùng thuộc một lớp)… Mỗi lĩnh vực khác nhau có các cách để tính độ tương ngữ nghĩa khác nhau.

* 1. **Thuật toán**

Với đề tài xây dựng chương trình mô phỏng tính toán độ tương tự giữa hai đối tượng, chúng em lựa chọn

* Đối tượng nghiên cứu: Văn bản (tiếng anh, tiếng Việt)
* Thuật toán sử dụng: Phương pháp TF-IDF và Cosine Similarity.
* Ngôn ngữ lập trình: Java
* Công cụ lập trình: Netbeans

### **2.2.1 Phát biểu bài toán**

Bài toán: Tính toán độ tương tự giữa hai đối tượng (văn bản)

Input: Hai văn bản

Output:Độ đo tương tự giữa 2 đối tượng (văn bản)

* + 1. **Giải quyết bài toán**
* **Tính toán độ tương tự văn bản**

**-Bước 1:** Tính toán giá trị Term Frequency (TF)

Term Frequency (TF) thể hiện mức độ quan trọng của một từ trong văn bản, hay TF là tần số xuất hiện của một từ trong một văn bản.

Trong thực tế, các văn bản có kích thước (tổng số từ) khác nhau. Trong các văn bản có kích thước lớn, tần số xuất hiện của một từ có thể sẽ nhiều hơn so với các văn bản có kích thước nhỏ hơn. Do đó, chúng ta cần phải Normalize (chuẩn hóa) tần số xuất hiện của một từ trong một văn bản dựa trên kích thước của văn bản đó. Đơn giản là chúng ta chia số lần xuất hiện của một từ cho tổng số từ trong văn bản.

TF(t) = n\_t/ N\_t

Trong đó

n\_t: tần số xuất hiện của một từ

N\_t: tổng số từ trong văn bản

**-Bước 2**: Tính toán giá trị Inverse Document Frequency (IDF- tần số nghịch của một từ trong một dataset)

Inverse Document Frequency (IDF) để giảm độ quan trong của những từ phổ biến và ít quan trọng như: trong tiếng Anh là *the, a, he, she,…*; đồng thời tăng độ quan trọng của những từ **“quan trọng”**.

IDF (t) = Log\_e (N\_d/ n\_d)

Trong đó:

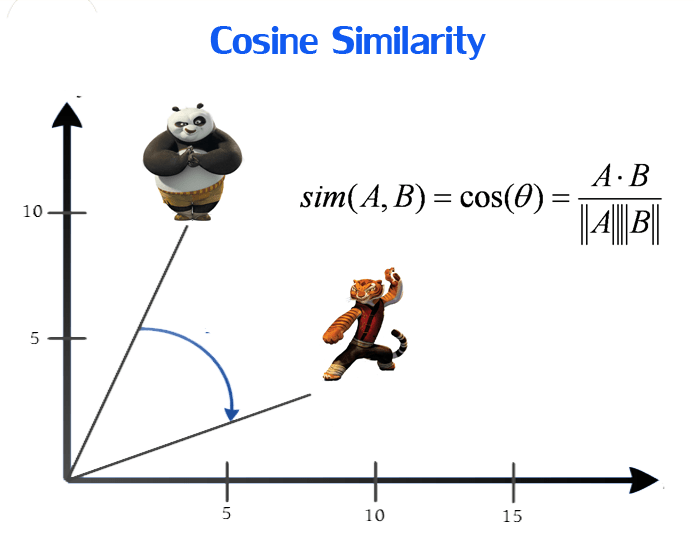
N\_d: Tổng số văn bản

n\_d: Số văn bản chứa từ cần tìm kiếm

**-Bước 3:** Tính toán giá trị TF\*IDF

-**Bước 4:** Vector Space Model –Cosine Similarity

Chúng ta mô tả mỗi document như là một vector. Một dataset được xem như là một tập hợp các vector trong một không gian vector. Mỗi từ trong không gian vector sẽ có trục của riêng nó. Bằng công thức sau:



### **2.2.3 Mã nguồn**

**- Tính tf**

private void readFileAndPreProcess() {

try {

Scanner in = new Scanner(new File(filename));

System.out.println("Reading file: " + filename + " and preprocessing");

while (in.hasNext()) {

String nextWord = in.next();

String filteredWord = nextWord.

replaceAll("[^A-Za-zđĐáàảãạăắằẳẵặâấầẩẫậđéèẻẽẹêếềểễệíìỉĩịóòỏõọôốồổỗộơớờởỡợúùủũụưứừửữựýỳỷỹỵ0-9]", "").

toLowerCase();

if (!(filteredWord.equalsIgnoreCase(""))) {

if (termFrequency.containsKey(filteredWord)) {

int oldCount = termFrequency.get(filteredWord);

termFrequency.put(filteredWord, ++oldCount);

} else {

termFrequency.put(filteredWord, 1);

}

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

**-Tính idf**

public double getInverseDocumentFrequency(String term) {

if (invertedIndex.containsKey(term)) {

double size = documents.size();

Set<Document> list = invertedIndex.get(term);

double documentFrequency = list.size();

return Math.log(size / documentFrequency);//log\_e

} else {

return 0;

}

}

**-Tính tf-idf**

private void createTfIdfWeights() {

System.out.println("Creating the tf-idf weight vectors");

Set<String> terms = corpus.getInvertedIndex().keySet();//ds term

for (Document document : corpus.getDocuments()) {

HashMap<String, Double> weights = new HashMap<>();

System.out.println("\nDocument: "+document.getFilename());

for (String term : terms) {

double tf = document.getTermFrequency(term);

double idf = corpus.getInverseDocumentFrequency(term);

double weight = tf \* idf;

weights.put(term, weight);

}

tfIdfWeights.put(document, weights);

}

}

**-Tính cosine similarity**

/\*\*

\* This method will return the magnitude of a vector.

\*

\* @param document the document whose magnitude is calculated.

\* @return the magnitude độ lớn của vector

\*/

private double getMagnitude(Document document) {

double magnitude = 0;

HashMap<String, Double> weights = tfIdfWeights.get(document);

for (double weight : weights.values()) {

magnitude += weight \* weight;

}

return Math.sqrt(magnitude);

}

/\*\*

\* This will take two documents and return the dot product.

\*

\* @param d1 Document 1

\* @param d2 Document 2

\* @return the dot product of the documents

\*/

private double getDotProduct(Document d1, Document d2) {

double product = 0;

HashMap<String, Double> weights1 = tfIdfWeights.get(d1);

HashMap<String, Double> weights2 = tfIdfWeights.get(d2);

for (String term : weights1.keySet()) {

product += weights1.get(term) \* weights2.get(term);

}

return product;

}

/\*\*

\* This will return the cosine similarity of two documents. This will range

\* from 0 (not similar) to 1 (very similar).

\*

\* @param d1 Document 1

\* @param d2 Document 2

\* @return the cosine similarity

\*/

public double cosineSimilarity(Document d1, Document d2) {

return getDotProduct(d1, d2) / (getMagnitude(d1) \* getMagnitude(d2));

}

## **2.3 Xây dựng chương trình**

### **2.3.1 Giao diện chính**

### **2.3.2 Hướng dẫn sử dụng**

# TÀI LIỆU THAM KHẢO

* http://tfidf.com/
* <http://luanvan.net.vn/luan-van/luan-van-tinh-toan-do-tuong-tu-ngu-nghia-van-ban-dua-vao-do-tuong-tu-giua-tu-voi-tu-53112/>
* http://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/