УДК 004.912

СРАВНЕНИЕ ФУНКЦИЙ ДЛЯ ИЗМЕРЕНИЯ СХОДСТВА МЕЖДУ ТЕКСТОВЫМИ ДОКУМЕНТАМИ

Бакиев Б., *старший преподаватель Университета им. С.Демиреля*
Мусабаев Р.Р. *Институт проблем информатики и управления МОН РК*

Түйін

Мәтіндік құжаттардың ұқсастығын өлшеу идеясы, көптеген бағыттарда, әсіресе берілген мәтінді алу және дайындау секторында кең резонансқа ие. Берілген мәтінді сандық векторда өрнектеу – үлкен еңбекті қажет етеді. Себебі біз сөздерді белгілеу, тоқта-сөздерді фильтрациялап, түбір сөздерге айналдыру, ал ең соңында TF-IDF қызметі бойынша есептеу жүргіземіз. Бүгінде құжаттардың ұқсастықтарын өлшеу туралы көптеген зерттеулер бар, бірақ көбіне Джакарт, Косинус, Дайс, Овэрлап қолданылып келеді.

Түйін сөздер: TF-IDF, Косинус, Оверлэп, Дайс, Джакарт

Summary

The idea of measuring the similarities of textual documents received a considerable attention in a number of areas, particularly in the area of Information retrieval. Converting the data into numerical vectors is not an easy task, because we use their layout of words and filtering stop words, and turn to the root word, and the finally is the calculation by TF-IDF function. Today, there are many researches on measuring the document similarities, but the most popular are the Jaccard, Cosine, Dice, and Overlap.

Keywords: TF-IDF, Cosine, Overlap, Dice, Jaccard

Аннотация. *Идея измерения сходства текстовых документов получила широкий резонанс в ряде направлений, в частности, в секторе «разработка и извлечение данных». Преобразовать данные в числовые векторы – скрупулезный труд, поскольку мы используем и разметки слов, и фильтрацию стоп-слов, и превращения в корневые слова, а окончательным звеном является расчет по функции TF-IDF. На сегодняшний день существуют результаты исследований многих идей по измерению сходства, но большей популярностью пользуются Джакарт, Косинус, Дайс, Овэрлап. Цель работы: сравнить все известные нам функции для поиска дополнительного сходства текстовых документов. В заключение анализа, на основе полученных результатов выведем градацию и ранжирование аналогий.*

Введение

21 век отличается тем, что на смену рукописной системе приходит электронный аналог. С введением новых прогрессивных технологий повышается качество, эффективность, производительность и конкурентоспособность независимо от сферы и рода деятельности. Новый век

подарил человечеству возможность найти «ЧТО? ГДЕ? КОГДА?», пользуясь услугами своего рода «разведчиков-поисковиков»: Google или Yahoo!

Существует достаточное количество приложений, которые предлагают решения задач поиска по сходству. Базовому сообществу хорошо известны проблемы, возникающие на пути решения при использовании объемных данных, апеллирования всевозможными функциями. Решение – задача не из легких. Структура статьи: во-первых, краткое описание, как переносить данные в числовые векторы, где использовать разметки, фильтрации стоп-слов, корьневые слова, и функции TF-IDF. Во-вторых, мы кратко остановимся на описании этих функциях и попробуем использовать их впрограмме Rapid-miner. Для наборов данных возьмем различные тексты с сайта популярного агентства новостей «Рейтер» размером около 20 мегабайт.

И в заключении попытаемся сравнить эти функции, произведя ранжирование в соответствии с их результатами.

1. Преобразовывание из текста в векторы

Чтобы обработать текст, мы должны сначала обработать его в форму, которая произведя анализ, укажет, что можно использовать.

Первым шагом в исследовании текста является сбор данных (то есть соответствующие документы). Во многих сценариях анализа текста, соответствующие документы уже предоставлены или являются частью описания проблемы. Например, поиск веб-приложения для интрасети не явно определяет соответствующие документы, чтобы существовать подобной веб-странице в интрасети. Если документы легко идентифицировать, то они могут быть получены, и главная задача – отредактировать образцы до высокого уровня. Человеческий фактор может нарушить целостность процесса сбора документов, а значит, надо быть особенно осторожным и внимательным. Порой, документы могут быть получены из архива документов либо из определенной информативной базы. В этих случаях, разумно ожидать, что очистка данных было сделано до их размещения, и мы можем быть уверены в качестве этих документов. На исследования и разработку текста используются методы, привлекающие более общие данные. Это обычно называется корпус.

После того как документы собраны, их следует преобразовать в единый формат, ведь в зависимости от того как получен тот или иной документ и формат варьируется. Например, в некоторых документах может быть создан собственный формат путем текстового процессора, другие, возможно, были созданы с помощью простого текстового редактора и сохранились как ASCII текст, но случается, что текст был отсканирован и сохранен в виде изображения. Очевидно, что если мы желаем обработать все документы, то обработать их следует в стандартном формате. Вот почему мы используем здесь коллекцию

Рэйтерс (Reuters) новостей, называемую Reuters RCV1 корпус, который можно извлечь с веб-сайта корпорации «Reuters».

1.1 Удаление символов (Tokenization)

Первым шагом в обработке текста является очистка от всех пометок, либо громоздких распространенных несогласованных предложений, поскольку данная процедура имеет фундаментальное значение для дальнейшего анализа. Без выявления метки, трудно представить путь извлечения более полной информации из документа. Символы пробела, табуляции не считаются словами. Они часто называются пробелом. Символы «( ) < > ! ? "» это разделители, которые являются своего рода жетоны символов : - . " и они не всегда могут быть разделителями. Все зависит от контекста.

Точка, запятая или двоеточие между цифровыми номерами обычно не считается разделителем. Любая другая запятая или двоеточие является разделителем и может быть маркером. Период может быть частью аббревиатуры (например, если она имеет или состоит из стечения прописных букв). Апостроф также имеет несколько применений. Когда он находится до и после слов, то должен рассматриваться как часть текущего маркера (например D'Angelo) . Если за ним следует однозначный ограничитель, это может быть закрытие внутренней цитаты или указание на притяжательные падеж. Апостроф предшествует ограничителю, так что можно различить внутренние и внешние кавычки. Знак «тире». Тире между двумя числами может быть символом вычитания, разделителя либо дефиса (например, 555-1212 как телефонный номер). Тире может находится и внутри конструкции, тогда следует ставить тире с обеих сторон. Чтобы добиться высокого уровня, следует настроить токенизаторна данный контекст иначе может потребоваться дополнительные действия для анализа тексати конечно же, незапланированной количество времени. Сосредоточимся на документации на английском языке. В других языках детали будут отличаться, хотя принцип общий.

2 Фильтрация коротких слов (FilteringStop-words).

Очень часто некоторые обще употребительные слова нужно исключить из лексикона, дабы они не несли определенную ценность. Подобные слова называют стоп-словами.

Пример топ-слова показано на рисунке-1.

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Рисунок-1. Стоп лист из 25 слов которые испольуются чаще всего в Reuters-RCV1.

Использование стоп-слов значительно снижает число входящих для регистраций, которые должны автоматически сохраняться системой. Стоп-слова вещь вполне безобидная. Не очень практичен и удобен поиск по ключевым словам с терминами. Некоторые названия песен и частей стихотворения построены целиком на стоп-словах ***(****Например: To be or not to be, Let It Be, I don't want to be ...).* Но, для лучшего результата следует удалять стоп-слова. Кроме того, очищая текст от подобных словесных конструкций, мы сокращаем время работы анализа.

2.1 Лемматизация (Lemmatization)

Как только персональный поток разделили на последовательные лексемы, каждый из маркеров необходимо привести к стандартному виду. Такой процесс называют лемматизацией. Сходство в документах может давать лишь незначительное преимущество и то не всегда. Обратим внимание, что одинь эффект, вытекающий из уменьшения числа различных типов в тексте слов может увеличить частоту встречаемости отдельных типов. Например, два одинаковых слова«Types» будет сокращен до корневой морфемы «Type» и будет читаться как слово «Type», наряду с экземплярами слов «Type» и «Typed», таким образом, они все будут читаться одинаково после сокращение их до корня. Кроме того, корень слова «rebelled»это«rebel», но корень слова «belled» является «bell», хотяслова схожи по написанию, по аналогии с паронимами. Для сходство алгоритмов поиска, которые принимают во внимание частоту повторения, иногда может изменить ситуацию. В других сценариях дополнительной обработки, может не быть значительных успехов.
Поэтому мы используем здесь рабочий английский словарь, который будем сравнивать со словарем, содержащим исконно историческое и традиционное написание, где за основу будем брать корневую морфему, которая и хранится в базе данных словарного состава.

2.2 Генерация в векторы

Коллективный набор функций обычно называется словарем слов. Жетоны (слова) в словаре являются основой для создания электронных таблиц числовых данных, соответствующих документу коллекции. Каждая строка представляет собой определенный документ, и каждый столбец представляет собой определенную функцию. Таким образом, ячейка в таблице является измерением особенности (соответствующий столбец) для документа (соответствующей строке). После того, набор функций был определен, коллекции документов могут быть преобразованы в формат электронной таблицы, где каждый столбец в таблице соответствует функции. Вместо того, чтобы хранить все нули, лучше представлять таблицы в виде набора разреженных векторов, причем строка представляет собой список пар, один из элементов пары будет номером столбца, а другой элемент имеет соответствующие ненулевым особенностям значения.Если не сохранять нули, то архив памяти может достигать огромных

размеров. Обработка программы может быть легко адаптирована для обработки этого формата. На рисунке-2 приведен несложный пример того, как электронная таблица превращается в разреженные векторы.

Таблица				Разреженные вектора
0	15	0	3	(2,15) (4,3)
12	0	0	0	(1,12)
8	0	5	2	(1,8) (3,5) (4,2)

Рис.2. Генерация таблицы в разреженных векторов

Следующим шагом после подсчета частоты слова в документе является изменение количества по воспринимаемому значению этого слова. Известныйкак TF-IDF формулировка была использована для вычисления весов или множества слов.

2.2.1 Частотатермина. (Term frequency)

TF-частота терминов, определяется как:

$$tf_{i,j}=\frac{n_{i,j}}{\sum_k n_{k,j}}$$

Где $n_{i,j}$ число терминов (t_i) встречающие в документе d_j, а знаменателем является сумма количество всех терминов в документе d_j, то есть размер документа будеть $|d_j|$.

2.2.2 Обратныечастотыдокумента (Inverse document frequency)

Обозначая, как обычно, общее количество документов в коллекции от D, определим обратную документа частоту (idf) от терминов t_i:

$$idf_i= \log_2 \left(\frac{|D|}{\{d:t_i\in d\}}\right),$$

Где,

$|D|$: количество всех документов

$\{d:t_i \in d\}$: количество тех документов где встречаются термины t_i. Если термина нет в документе, то это принесет нас к делению на ноль. Поэтому в формуле (idf) следует использовать

$1+ |\{d:t_i \in d\}|$ вместо $\{d:t_i \in d\}$.

2.2.3 tf-idf

Теперь объединим определения частоты термина (TF) и обратной частоты документа (IDF) для получения композитных вес для каждого термина в каждом документе.

$$(tf-idf)_{i,j}=tf_{i,j}\times idf_i \qquad (1)$$

а) Пример для пойска одного слова: Рассмотрим документ, содержащий 100 слов, в котором слово «корова» появляется 3 раза. Термин частота (TF) для слова «корова» будет выглядеть так: (3/100) = 0,03. Теперь предположим, что у нас есть 10 миллионов документов, и слово «корова» появляется в 1000 из них. Теперь обратной частоты документа (IDF) вычисляется так: (10 000 000/1 000) = 4.

Тогда TF-IDF будет равен: 0,03 × 4 = 0,12.

б) Другой пример для пойска нескольких слов или одного предложения из нескольких слов: Скажем пойсковый текст (Q) или документ являетсятак "gold silver truck".

И база данных имеет три документа (D = 3) стекстами:

D1: "Shipment of gold damaged in a fire"

D2: "Delivery of silver arrived in a silver truck"

D3: "Shipment of gold arrived in a truck"

Давайте без фильтраций и лемматизаций поищем текст Q в документах D_i,(i=1,2,3). Появиться вот такая таблица:

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$											
Query, Q: "gold silver truck" D_1: "Shipment of gold damaged in a fire" D_2: "Delivery of silver arrived in a silver truck" D_3: "Shipment of gold arrived in a truck" D = 3; IDF = log(D/df_i)											
		Counts, tf_i						Weights, $w_i = tf_i * IDF_i$			
Terms	Q	D_1	D_2	D_3	df_i	D/df_i	IDF_i	Q	D_1	D_2	D_3
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

Теперь, в предыдущей таблице для каждого документа (D) и запроса (Q), вычислим все длины векторов (нулевые члены игнорируются).

Получится следующее:

$$|D_1| = \sqrt{0.4771^2 + 0.4771^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.5173} = 0.7192$$

$$|D_2| = \sqrt{0.1761^2 + 0.4771^2 + 0.9542^2 + 0.1761^2} = \sqrt{1.2001} = 1.0955$$

$$|D_3| = \sqrt{0.1761^2 + 0.1761^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.1240} = 0.3522$$

$$\therefore |D_i| = \sqrt{\sum_i w_{i,j}^2}$$

$$|Q| = \sqrt{0.1761^2 + 0.4771^2 + 0.1761^2} = \sqrt{0.2896} = 0.5382$$

$$\therefore |Q| = \sqrt{\sum_i w_{Q,j}^2}$$

Затем мы вычисляем скалярное произведение (ноль игнорируется).

$$Q \bullet D_1 = 0.1761 * 0.1761 = 0.0310$$

$$Q \bullet D_2 = 0.4771 * 0.9542 + 0.1761 * 0.1761 = 0.4862$$

$$Q \bullet D_3 = 0.1761 * 0.1761 + 0.1761 * 0.1761 = 0.0620$$

$$\therefore Q \bullet D_i = \sum_i w_{Q,j} w_{i,j}$$

Теперь находим сходства спомощю Косинус функций:

$$\text{Cosine } \theta_{D_1} = \frac{Q \bullet D_1}{|Q| * |D_1|} = \frac{0.0310}{0.5382 * 0.7192} = 0.0801$$

$$\text{Cosine } \theta_{D_2} = \frac{Q \bullet D_2}{|Q| * |D_2|} = \frac{0.4862}{0.5382 * 1.0955} = 0.8246$$

$$\text{Cosine } \theta_{D_3} = \frac{Q \bullet D_3}{|Q| * |D_3|} = \frac{0.0620}{0.5382 * 0.3522} = 0.3271$$

$$\therefore \text{Cosine } \theta_{D_i} = \text{Sim}(Q, D_i)$$

$$\therefore \text{Sim}(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}}$$

Теперь, получив результаты, мы можем сказать, что документ (D2) в большей степени сходится с документом (Q), которую мы искали. Таким образом, мы можем поставить их по убыванию сходства:

1) D2=0.8246
2) D3=0.3271
3) D1=0.0801

Вот так на данный момент и работают все поисковые вэб-сайты.

3. **Функций, находящие сходства**

3.1 Косинус измерение:

$$\mathrm{Cos}\,\theta = C(\vec{X},\vec{Y}) = \frac{\vec{X}\cdot\vec{Y}}{\|\vec{X}\|\cdot\|\vec{Y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \qquad (3.1)$$

Если угол между X и Y равень нулю, то эти документы полностью сходятся, так как $\mathrm{Cos}(0) = 1$. Если угол между X и Yравень 90, то эти документы не сходятся совсем, потому что $\mathrm{Cos}(90) = 0$.

Ответ всегдабудет лежать между 0 и 1, иначе отрицательные числа документов не имеют смысла.

3.2. Джакарт измерение:

$$J=J(\vec{X},\vec{Y}) = \frac{\vec{X}\cdot\vec{Y}}{\|\vec{X}\|^2+\|\vec{Y}\|^2-\vec{X}\cdot\vec{Y}} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2+\sum_{i=1}^{n} y_i^2-\sum_{i=1}^{n} x_i y_i} \qquad (3.2)$$

3.3. Дайс измерение:

$$D = D(\vec{X},\vec{Y}) = \frac{2\vec{X}\cdot\vec{Y}}{\|\vec{X}\|^2+\|\vec{Y}\|^2} = \frac{2\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2+\sum_{i=1}^{n} x_i^2} \qquad (3.3)$$

Обобщённая форма Дайса:

$$D_\alpha = D_\alpha(\vec{X},\vec{Y}) = \frac{\vec{X}\cdot\vec{Y}}{\alpha\|\vec{X}\|^2+(1-\alpha)\|\vec{Y}\|^2} = \frac{2\sum_{i=1}^{n} x_i y_i}{\alpha\sum_{i=1}^{n} x_i^2+(1-\alpha)\sum_{i=1}^{n} x_i^2} \qquad (3.3a)$$

3.4 Оверлап измерение:

$$O_1(\vec{X},\vec{Y}) = \frac{\vec{X}\cdot\vec{Y}}{\min(\|\vec{X}\|^2,\|\vec{Y}\|^2)} = \frac{\sum_{i=1}^{n} x_i y_i}{\min(\sum_{i=1}^{n} x_i^2,\sum_{i=1}^{n} y_i^2)} \qquad (3.4a)$$

И еще одна форма Оверлапа:

$$O_2(\vec{X},\vec{Y}) = \frac{\vec{X}\cdot\vec{Y}}{\max(\|\vec{X}\|^2,\|\vec{Y}\|^2)} = \frac{\sum_{i=1}^{n} x_i y_i}{\max(\sum_{i=1}^{n} x_i^2,\sum_{i=1}^{n} y_i^2)} \qquad (3.4б)$$

Из (3.4а) и (3.4б) ясно что $O_1(\vec{X},\vec{Y}) > O_2(\vec{X},\vec{Y})$.

4. **Сравнение функции**

С помощью функций, которые мы обсуждали, сравним текстовые документы из данных агентства новостей Рейтер.

4.1. Отношение (Relations)

Эта функция делит 2D координат с одной диагонали. Эта функция может измерять только две функции. Каждая точка, которая лежит в оси результат, полученной от сходства двух документов. В зависимости от координат точки результатов можно анализировать их таким образом, если точка результат лежит вблизи оси Y, то функция (сходство функции), которая находится на оси Y дает более высокий результат, в противном случае, если в результате точка лежит вблизи X оси, то функция по оси X дает более высокий результат, чем функция на оси Y. Если точка лежит на диагонали, то обе функции являются одинаковыми. В зависимости от того, сколько точек лежат и в какую сторону по диагонали, мы можем сказать, какая из функции покажет число больше или меньше по сравнению с другой функцией.

Первым мы сравним косинус и оверлап, с помощью популярной приложений Rapid-miner.

Рисунок-3 показывает, что оверлап дает лучший результат, и только некоторые лежат на стороне косинус, а все остальные на сторонеоверлап. В немногих результатах документов косинус дает лучший результат, чем оверлап. Но в целом результаты документововерлапна порядок выше, чем косинус.

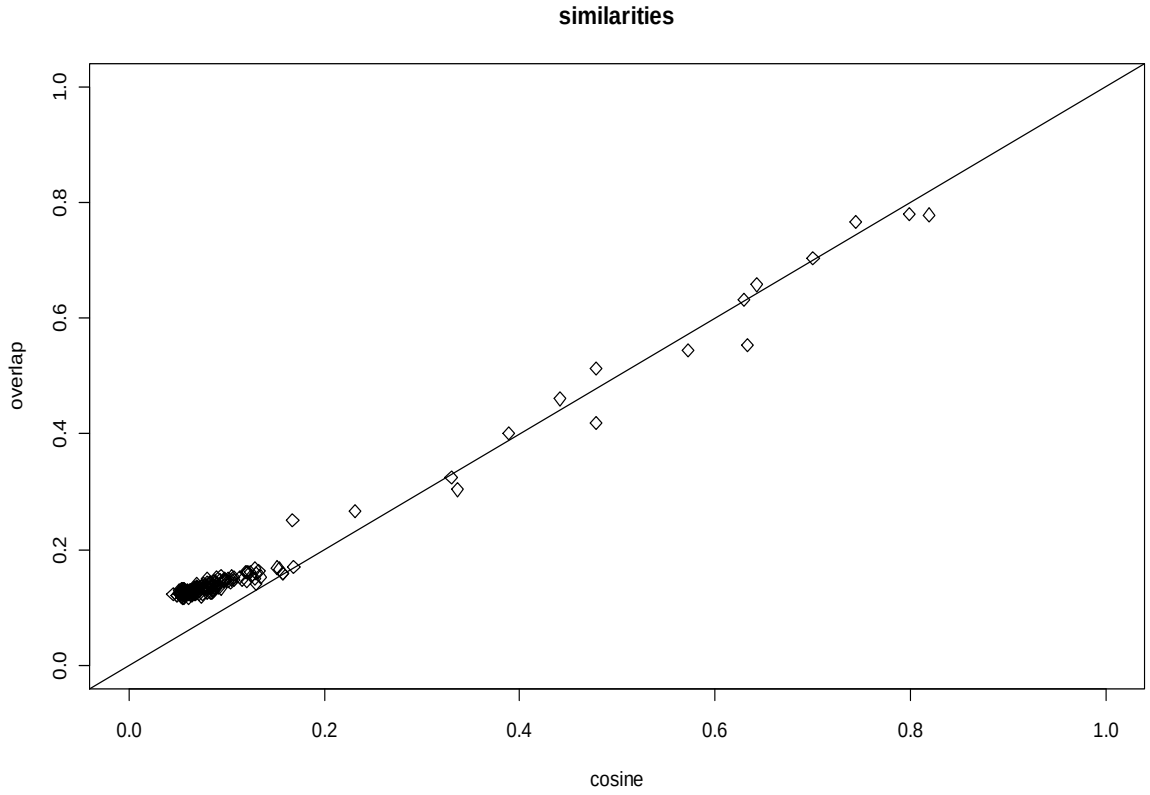

Рис.3.Отношение между косинус и оверлап

Есть и другие приложений, которые можно использовать для анализа. Обратимся к Excel. Рисунок-4 показывает преводсходство Оверлапа над другими функциями. Только в некоторых результатах документов Косинус превосходит остальные. Но опять же в целом, результаты документов Оверлап показывает свое доминирующее положение. По предыдущему анализу и в данном исследовании становится очевидным превосходство Оверлап (a) над остальными функциями.

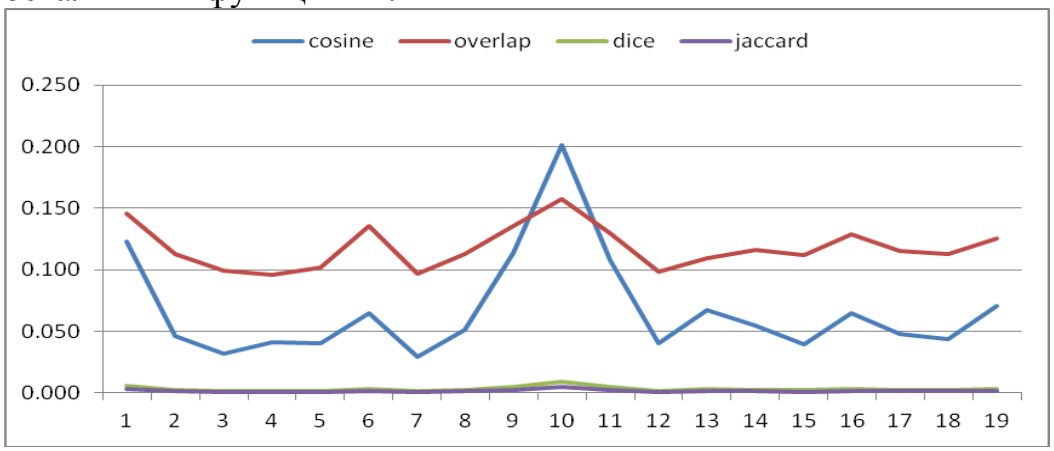

Рис.4. Анализ функций косинус, оверлап, дайс и джакарт

По рисунку 4, трудно разглядеть, который из дайс и джакарт покажет наибольший результат. Поэтомусразу передем к их сравнению. Расмортим их вотношениях (Relations). Рисунок-5 четко показывает, что Дайс дает результат лучше, чем Джакарт, так как все точки лежат на стороне Дайс.

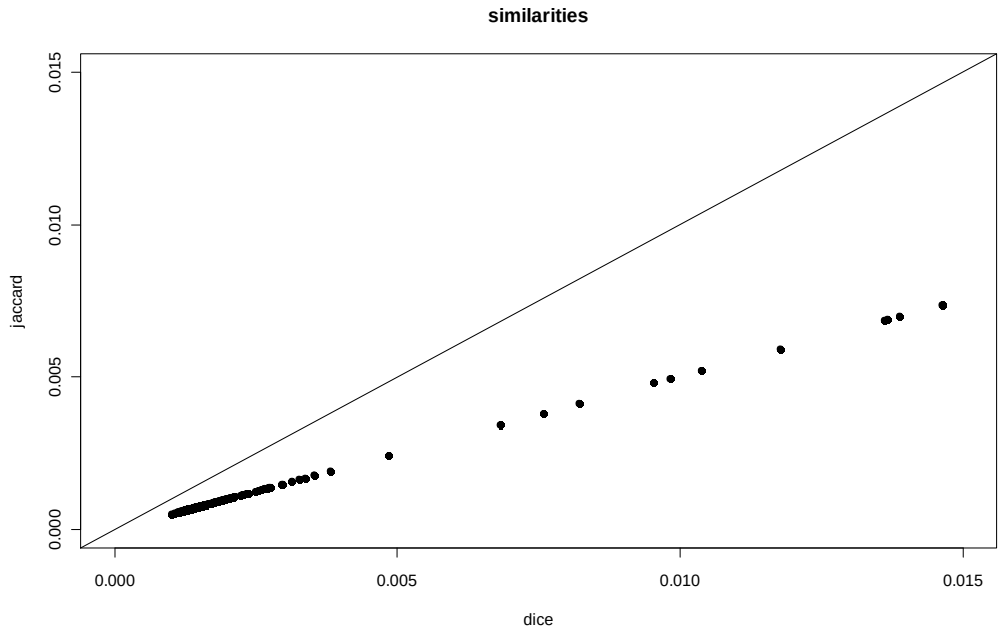

Рис.5.Отношение между Джакарт и Дайс

Мы можем утверждать, что Оверлап дает лучший результат, чем остальные функции, которые мы использовали ранее. Итак: Оверлап > Косинус > Дайс > Джакарт

5 Заключение

Результаты эксперимента показали, чтомежду пятью функциями в текстовых документах сходства функции Оверлап дает превосходящие результаты, поскольку мы пришли к выводу:

Оверлап > Косинус > Дайс > Джакарт

Вся работа была проделана, на основе поиска схожести в словах, исключая синонимы и многозначные слова. В документах слова могут быть одинаковы по смыслу и в то же время являться совершенно разными.

В перспективе мы планируем произвести анализ сходства документов по лексическому значению (семантике) слов.

Литература

1. Bakiev B., “Text based document similarity measure”, IKECCO 2011, pp. 47-56.
2. Sholom M. Weiss, Nitin Indurkhya, Tong Zhang, Fred J. Damerau, “Text mining: Predictive Methods for Analyzing Unstructured Information”, Springer, USA, 2005, pp. 1-2, 15-25, 85-89.
3. Leo Egghe, “New relations between similarity measures for vectors based on vector norms”, Journal of the American Society for Information Science and Technology, New York, USA, 2009, vol. 60, pp. 2-6.
4. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, “Introduction to Information Retrieval”, Cambridge University Press, USA, 2008, pp. 117-119.