Nama : Rini Jannati

NPM : 1706005905

Tugas 3

Implementasi model retrieval

Hal yang pertama kali saya lakukan adalah membaca dokumen koleksi.txt dan melakukan pemisahan dokumen dan melakukan tokenisasi per dokumen. Algoritma yang saya terapkan adalah

1. Buka dokumen koleksi.txt
2. Baca dokumen tersebut perbaris
3. Ketika ada tag <NO> maka baris tersebut dihapus tag <NO> dan </NO> serta spasinya sehingga hanya tinggal nomor dokumennya lalu nomor dokumen tersebut di push ke array dokumen dan juga disimpan dalam sebuah variabel yang bernama number
4. Jika tidak maka masuk ke bagian untuk mendapatkan teks bacaannya. Ketika baris memiliki tag <JUDUL> serta bukan merupakan <TEKS>, </TEKS>,<DOK> dan </DOK> maka baris langsung menghapus kata <JUDUL> dan </JUDUL> lalu dilakukan tokenisasi, kata juga langsung dilakukan stemming dengan file yang ada pada TUGAS 2. Kata-kata tersebut masuk kedalam Hash yang dibuat dengan menggunakan struktur HASH of HASH sehingga hash menyimpan nomor dokumen, hasil token per dokumen dan hitungan token per dokumen. Hasil token juga dipush ke sebuah array word untuk koleksi.

Hasil Koding:

```
#ambil koleksi kata dari koleksi dokumen:
open (NEWFILE,"koleksi.txt") or die ("Can't open the file");
while($lines = <NEWFILE>){
	chomp($lines);
	#Mengambil nilai nomor dokumen pada koleksi
	if(grep(/<NO>/,$lines)){
		$lines =~ s/<NO>//;
		$lines =~ s/<\/NO>//;
		$lines =~ s/\s//;
		$number = $lines;
		push @doc, $number;
	}else{
		#mengambil nilai teks dan judul pada koleksi
```

```
            if(grep(/<JUDUL>/,$lines)    ||    (grep(!/<TEKS>/,$lines)    &&
grep(!/<\/TEKS>/,$lines)            &&            grep(!/<\/DOK>/,$lines)&&
grep(!/<DOK>/,$lines))){
                  $lines =~ s/<JUDUL>//;
                  $lines =~ s/<\/JUDUL>//;
                  $lines =~ tr/[A-Z]/[a-z]/;
                  $lines =~ tr/-!?@#$%^&*()~`[]{}:;""'',”<>,._+=\|/ /;
                  my @key = split " ", $lines;
                  @key = grep (!/[^a-z0-9]/,@key);
                  foreach(@key){
                        if(length($_)>4){
                              $_ = cekimbuhan($_);
                              $_ = ceksisipan($_);
                        }
                        #hitung katanya per kata dalam 1 dokumen
                        $document{$_}{$number}++;

                  }

                  push @word, @key;

            }
      }
}
close(NEWFILE);
```

Setelah melakukan tokenisasi, saya melakukan tokenisasi untuk seluruh kata di koleksi dengan menghitung banyak kata pada array kata dan jumlah kata pada koleksi.

```
#jumlah kata dalam koleksi:
$countkata{$_}++ foreach(@word);
#menghitung banyak kata pada koleksi
foreach my $w (sort keys %countkata){
      $TerminCollection += $countkata{$w};
}
```

Setelah itu saya melakukan proses yang dibutuhkan untuk menghitung cossine similarity. Menghitung IDF dan IDF Saya melakukan normalisasi bentuk data untuk membentuk sebuah data matriks [term][dokumen] dengan tetap menggunakan struktur data hash. Hasil data dapat dilihat pada file hasilTF.csv dan hasilTFIDF.csv.

```
my $NDoc = @doc;
my %totalKuadrat;
$totalKuadrat{$_} = 0 foreach(@doc);
#TF-IDF kata
my %totalKuadratIDF;
$totalKuadratIDF{$_} = 0 foreach(@doc);
foreach my $w (sort keys %countkata){
      #mencari IDF per kata
      my $nDTF = keys %{$document{$w}};
      $IDFofWord{$w} = log($NDoc/$nDTF);
```

```
	#normalisasi dokumen, ketika kata ada pada dokumen maka frekuensi
kata = TF, jika tidak maka kata pada dokumen = 0
	foreach my $d (@doc){
		if($document{$w}{$d}){
			$docoffile{$w}{$d} = $document{$w}{$d};
		}else{
			$docoffile{$w}{$d}=0;
		}
		#menghitung vektor namun belum di kuadratkan
		$TerminDocument{$d} += $docoffile{$w}{$d};
		$TF_kuadrat{$w}{$d} = $docoffile{$w}{$d} ** 2;
		$totalKuadrat{$d} += $TF_kuadrat{$w}{$d};

		$IDFofDoc{$w}{$d} = $docoffile{$w}{$d} * $IDFofWord{$w};
		$TFIDF_kuadrat{$w}{$d} = $IDFofDoc{$w}{$d} ** 2;
		$totalKuadratIDF{$d} += $TFIDF_kuadrat{$w}{$d};

	}

}

foreach my $d (sort @doc){
	#panjang vektor per document
	$totalKuadrat{$d} = sqrt($totalKuadrat{$d});
	$totalKuadratIDF{$d} = sqrt($totalKuadratIDF{$d});

}
```

Mengambil nilai pada query:

```
#Query nilainya:
open(FILE,"query.txt") or die ("Can't open the file");
while($lines = <FILE>){
	my @array = split/\t/,$lines;
	$array[1] =~ tr/-!?@#$%^&*()~`[]{}:;""'',”<>,._+=\|/ /;
	$array[1] =~ tr/[A-Z]/[a-z]/;
	my @wordarray = split " ",$array[1];
	foreach(@wordarray){
		#$_ = stempart1($_);
		if(length($_)>4){
			$_ = cekimbuhan($_);
			$_ = ceksisipan($_);
		}
		$query{$array[0]}{$_}++;
		#print($_);
	}
	#inisialisasi 0 untuk nilai sigma q(w)*d(w)
	$sigma{$array[0]}{$_} = 0 foreach(@doc);
	$sigmaIDF{$array[0]}{$_} = 0 foreach(@doc);
}
```

Membuat soundex pada koleksi kata untuk query handling

```
#membuat soundex:
my %soundexkata;
foreach my $i (sort keys %countkata){
	if(grep/[a-z]/,$i){
		my $sound = soundex($i);
		$soundexkata{$i} = $sound;
		print File "$i,$sound\n";
	}
}
```

Query Handling dilakukan dengan cara mencocokkan hasil soundex dari tokenisasi kata query. Di program ini saya melakukan pengecekan kondisi jika kata pada query tidak terdapat pada koleksi kata maka kata pada query diubah ke soundex lalu dilakukan pengecekan pada soundex koleksi, jika ada kode soundex yang sama maka kata tersebut kemungkinan adalah kata koleksi. Namun ada sebuah kondisi yang memungkinkan adanya beberapa kata memiliki soundex yang sama, sehingga kita membutuhkan sebuah fungsi baru untuk mengecek kata yang memiliki kode yang sama tersebut juga memiliki beberapa kata yang mirip dengan query. Pada proses itu dibutuhkan proses pencocokan yang biasa disebut n-gram overlap dengan menggunakan bigram. Pemilihan hasil query perbaikannya adalah dengan memilih jaccard coefficient tertinggi sebagai kata query yang mungkin mirip dengan query yang perlu dikoreksi.

Koding untuk check kata

```
my %totalquery;
foreach my $q(sort keys %query){
	foreach my $w(sort keys %{$query{$q}}){
		#HITUNG BAGIAN QUERY
		#correction issue
		if(!$countkata{$w}){
			my %jcc;
			my @array;
			#cari soundex yang sama
			my $soundquery = soundex($w);
			foreach my $i (sort keys %soundexkata){
				if($soundquery eq $soundexkata{$i}){
					#mencari susunan kata yang sama
					$jcc{$i} = jaccard($w,$i);
				}
			}
			#ambil nilai skor jaccard dengan mengurutkan dari nilai
terbesar
			 @array = sort {$jcc{$b} <=> $jcc{$a}} keys %jcc;
			my $y = 0;
			foreach my $x (0..@array-1){
				$y = $array[0];
			}
			#hapus kata yang dikoreksi.
			delete $query{$q}{$w};
```

```
                        #ambil nilai terbesar dan lakukan perhitungan
                        $query{$q}{$y}++;
                }
        }
}
```

Koding N-gram overlap

```
sub jaccard{
        my ($inp1, $inp2) = @_;
        my $result;
        my $check =0;
        my @array1;
        my @array2;

        if(length($inp1) < 6 && length($inp1) eq length($inp2)){
                @array1 = split (//,$inp1);
                @array2 = split (//,$inp2);

                foreach my $i (0..@array1-1){
                        foreach my $j (0..@array2-1){
                                if ($array1[$i] eq $array2[$j]){
                                        $check++;
                                        $array1[$i] ='0';
                                        $array2[$j] = '1';
                                }
                        }
                }
                #print ($check);

        }else{
                @array1 = makebigram($inp1);
                @array2 = makebigram($inp2);


                foreach my $i (0..@array1-1){
                        foreach my $j (0..@array2-1){
                                if ($array1[$i] eq $array2[$j]){
                                        $check++;
                                }
                        }
                }
        }

        my $arr1 = @array1;
        my $arr2 = @array2;

        $result = $check / ($arr1+$arr2-$check);
        return $result;
}

sub makebigram{
        my ($inp) =@_;
        my @bigram;

        foreach my $i (0..length($inp)-1){
                $k = substr ($inp,$i,2);
                push @bigram, $k;
        }
        return @bigram;
}
```

Setelah itu query dikalikan IDFnya agar dapat kata yang dapat dicari cossine similarity dengan menggunakan IDF

```
#QUERY
my %totalqueryIDF;
foreach my $q(sort keys %query){
	foreach my $w(sort keys %{$query{$q}}){
		#QUERY
		$totalquery{$q} += $query{$q}{$w} ** 2;
		#IDF QUERY
		$IDFofQuery{$q}{$w} = $query{$q}{$w} * $IDFofWord{$w};
		$totalqueryIDF{$q} += $IDFofQuery{$q}{$w} ** 2;
	}
	$totalquery{$q} = sqrt ($totalquery{$q});
	$totalqueryIDF{$q} = sqrt ($totalqueryIDF{$q});
}
```

Setelah mendapatkan nilai TF dan TFIDF pada dokumen, panjang vektor dokumen, nilai TF dan IDF pada query, dan panjang query maka program dapat menghitung nilai cossine similarity. Program juga bisa menghitung unigram language model dengan mixture model karena sudah mendapatkan nilai banyak sebuah kata dalam dokumen, banyak kata per dokumen, banyak sebuah kata di koleksi dan total seluruh kata di koleksi. Maka kodingan dapat dilihat seperti:

```
#perkalian cossine similarity

open(File,"> querysimilarity.txt") or die ("can't open file");
foreach my $q(sort keys %query){
	foreach my $d(sort @doc){
		#mencari total perkalian query dan document
		foreach my $w(sort keys %{$query{$q}}){
		#sigma Q(w) * D(w)
			my $x = $query{$q}{$w} * $docoffile{$w}{$d};
			$sigma{$q}{$d} += $query{$q}{$w} * $docoffile{$w}{$d};
			my $y = $IDFofQuery{$q}{$w} * $IDFofDoc{$w}{$d};
			$sigmaIDF{$q}{$d}	+=	$IDFofQuery{$q}{$w}	*
$IDFofDoc{$w}{$d};
		}
		#count all: dengan kondisi kata pada dokumen tidak 0.
		if($TerminDocument{$d} != 0){
			$sim{$q}{$d}	=	$sigma{$q}{$d}/($totalquery{$q}	*
$totalKuadrat{$d});
			print	File	"$sim{$q}{$d}	=
$sigma{$q}{$d}/($totalquery{$q} * $totalKuadrat{$d})\n";
			$simIDF{$q}{$d} = $sigmaIDF{$q}{$d}/($totalqueryIDF{$q}
* $totalKuadratIDF{$d});
			#unigram language model dengan mixture model
```

```
                        for(my $lmd=0.3; $lmd<0.9; $lmd+=0.1){
                              my $equal =1;
                              foreach my $w(sort keys %{$query{$q}}){
                                    my    $probabil    =    mixturemodel($lmd,
$docoffile{$w}{$d},$TerminDocument{$d},
$countkata{$w},$TerminCollection);
                                    #$equal *= $unigram{$lmd}{$q}{$w};
                                    $equal *= $probabil;
                              }
                              $unigram{$lmd}{$q}{$d} = $equal;

                        }
                  }
            }
            print File"\n";
}
#unigram language dengan mixture model
sub mixturemodel{
      my ($lamda, $TFword, $LD, $CFword, $TD) = @_;
      my $prob;
      #print"$TD\n";
      $prob = ($lamda * ($TFword/$LD)) + ((1-$lamda)*($CFword/$TD));

      return $prob;
}
```

Analisis:

Hasil seluruh nilai proses cossine similarity dengan nilai TF dapat dilihat pada sim.csv, cossine similarity dengan nilai TFIDF dapat dilihat pada simTF.csv dan unigram language dengan mixture model dapat dilihat pada unigram.csv

Hasil Top 5 cossine similarity TF:

Query	Rank	Document	Score
Q01	1	1106053552-17	0.477809467039376
Q01	2	1106053672-17	0.456677097615768
Q01	3	1106053552-12	0.406699489848499
Q01	4	1106053552-9	0.401607268919712
Q01	5	1106053552-6	0.379923415158478
Q02	1	1106001800-7	0.488860222000338
Q02	2	1106001800-4	0.479033328031948
Q02	3	1106047770-9	0.470248052883187

Q02	4	1106047770-10	0.465810814065513
Q02	5	1106053470-14	0.463334543372895
Q03	1	1106007104-15	0.479301293269411
Q03	2	1106007104-19	0.458130681061892
Q03	3	1106047764-10	0.429041295356308
Q03	4	1106087534-20	0.410477766304291
Q03	5	1106053634-018	0.408446806541952
Q04	1	1106001800-4	0.437117911829153
Q04	2	1106047770-14	0.387115431028666
Q04	3	1106047770-11	0.385708754668975
Q04	4	1106087660-11	0.370419603277933
Q04	5	1106053470-13	0.369494213792278
Q05	1	1106012003-5	0.642618569661179
Q05	2	1106005673-17	0.604435949977312
Q05	3	1106022603-4	0.551069272254633
Q05	4	1106087603-4	0.539886617035148
Q05	5	1106087603-13	0.528238586858023
Q06	1	1106053552-17	0.477809467039376
Q06	2	1106053672-17	0.456677097615768
Q06	3	1106053552-12	0.406699489848499
Q06	4	1106053552-9	0.401607268919712
Q06	5	1106053552-6	0.379923415158478

Hasil Top 5 pada perkalian cossine similarity dengan IDF

Query	Rank	Document	Score
Q01	1	1106053552-9	0.401607268919712
Q01	2	1106053552-6	0.379923415158478
Q01	3	1106053672-17	0.456677097615768
Q01	4	1106053672-4	0.379704249628178
Q01	5	1106053552-15	0.348045466408768
Q02	1	1106001800-6	0.457348120456344
Q02	2	1106004310-10	0.457348120456344
Q02	3	1106053470-12	0.387731312212205
Q02	4	1106053470-3	0.404868422720656
Q02	5	1106004310-15	0.457610044650721
Q03	1	1106007104-19	0.458130681061892
Q03	2	1106087534-20	0.410477766304291
Q03	3	1106007104-09	0.372717141042495
Q03	4	1106000874-16	0.327391620477899
Q03	5	1106047764-20	0.330841289063537
Q04	1	1106001800-4	0.437117911829153
Q04	2	1106087660-11	0.370419603277933
Q04	3	1106047770-14	0.387115431028666
Q04	4	1106001800-8	0.356465523119603
Q04	5	1106047770-11	0.385708754668975
Q05	1	1106012003-5	0.642618569661179
Q05	2	1106087603-13	0.528238586858023
Q05	3	1106005673-17	0.604435949977312
Q05	4	1106087603-4	0.539886617035148
Q05	5	1106022603-4	0.551069272254633

Q06	1	1106053552-9	0.401607268919712
Q06	2	1106053552-6	0.379923415158478
Q06	3	1106053672-17	0.456677097615768
Q06	4	1106053672-4	0.379704249628178
Q06	5	1106053552-15	0.348045466408768

Hasil top 3 pada model unigram language per lamda 0.3-0.9

Lamda	Query	Rank	Document	Score
0.3	Q01	1	1106023032-13	9.09316985803055e-13
0.3	Q01	2	1106053552-17	5.73530353115425e-13
0.3	Q01	3	1106053552-9	3.66446007587905e-13
0.3	Q02	1	1106001800-6	1.26498314260242e-11
0.3	Q02	2	1106004310-10	1.26498314260242e-11
0.3	Q02	3	1106004310-15	1.2490884824091e-11
0.3	Q03	1	1106007104-11	3.94861568858649e-10
0.3	Q03	2	1106000874-15	3.94861568858649e-10
0.3	Q03	3	1106000874-7	2.55831869655351e-10
0.3	Q04	1	1106053470-13	2.72903637664741e-11
0.3	Q04	2	1106004310-13	2.58432860859228e-11
0.3	Q04	3	1106053470-11	1.80362024376216e-11
0.3	Q05	1	1106012003-5	2.95767562729689e-06
0.3	Q05	2	1106005673-17	1.29528242140632e-06
0.3	Q05	3	1106005673-16	1.06463133705347e-06
0.3	Q06	1	1106023032-13	9.09316985803055e-13
0.3	Q06	2	1106053552-17	5.73530353115425e-13
0.3	Q06	3	1106053552-9	3.66446007587905e-13

0.4	Q01	1	1106023032-13	2.16208553933874e-12
0.4	Q01	2	1106053552-17	1.16697526371339e-12
0.4	Q01	3	1106053552-9	6.40621947035807e-13
0.4	Q02	1	1106004310-15	2.26736718375349e-11
0.4	Q02	2	1106001800-6	2.24261119537921e-11
0.4	Q02	3	1106004310-10	2.24261119537921e-11
0.4	Q03	1	1106000874-15	5.37194073309446e-10
0.4	Q03	2	1106007104-11	5.37194073309446e-10
0.4	Q03	3	1106000874-7	3.48456126029536e-10
0.4	Q04	1	1106053470-13	5.8925446063349e-11
0.4	Q04	2	1106004310-13	5.52740481279603e-11
0.4	Q04	3	1106053470-11	3.11002578500103e-11
0.4	Q05	1	1106012003-5	6.07239708231423e-06
0.4	Q05	2	1106005673-17	2.56351568591951e-06
0.4	Q05	3	1106005673-16	2.04959084683657e-06
0.4	Q06	1	1106023032-13	2.16208553933874e-12
0.4	Q06	2	1106053552-17	1.16697526371339e-12
0.4	Q06	3	1106053552-9	6.40621947035807e-13
0.5	Q01	1	1106023032-13	4.46006904815584e-12
0.5	Q01	2	1106053552-17	1.97640405215376e-12
0.5	Q01	3	1106053552-12	1.01547826801149e-12
0.5	Q02	1	1106004310-14	3.69488248279338e-11
0.5	Q02	2	1106004310-15	3.53811379316495e-11
0.5	Q02	3	1106001800-6	3.43282547293788e-11
0.5	Q03	1	1106000874-15	6.57773442567032e-10
0.5	Q03	2	1106007104-11	6.57773442567032e-10

0.5	Q03	3	1106000874-7	4.48235811781993e-10
0.5	Q04	1	1106053470-13	1.14558749021056e-10
0.5	Q04	2	1106004310-13	1.06669155452731e-10
0.5	Q04	3	1106053470-11	4.69059950305782e-11
0.5	Q05	1	1106012003-5	1.08383965839937e-05
0.5	Q05	2	1106005673-17	4.46817543099032e-06
0.5	Q05	3	1106005673-16	3.5038993300981e-06
0.5	Q06	1	1106023032-13	4.46006904815584e-12
0.5	Q06	2	1106053552-17	1.97640405215376e-12
0.5	Q06	3	1106053552-12	1.01547826801149e-12
0.6	Q01	1	1106023032-13	8.32750908008499e-12
0.6	Q01	2	1106053552-17	2.88540574386533e-12
0.6	Q01	3	1106053552-12	1.44018789705663e-12
0.6	Q02	1	1106004310-14	6.69360594027064e-11
0.6	Q02	2	1106001800-2	4.90039279796721e-11
0.6	Q02	3	1106004310-15	4.85631733369101e-11
0.6	Q03	1	1106007104-11	7.33250923398143e-10
0.6	Q03	2	1106000874-15	7.33250923398143e-10
0.6	Q03	3	1106000874-7	5.55170029869853e-10
0.6	Q04	1	1106053470-13	2.05740045571578e-10
0.6	Q04	2	1106004310-13	1.90442512694545e-10
0.6	Q04	3	1106001800-4	6.31815105792149e-11
0.6	Q05	1	1106012003-5	1.76062057313058e-05
0.6	Q05	2	1106005673-17	7.1385894524889e-06
0.6	Q05	3	1106005673-16	5.51855509644443e-06
0.6	Q06	1	1106023032-13	8.32750908008499e-12

0.6	Q06	2	1106053552-17	2.88540574386533e-12
0.6	Q06	3	1106053552-12	1.44018789705663e-12
0.7	Q01	1	1106023032-13	1.44346351103603e-11
0.7	Q01	2	1106053552-17	3.64880410933301e-12
0.7	Q01	3	1106053552-12	1.7805310915594e-12
0.7	Q02	1	1106004310-14	1.13803915745701e-10
0.7	Q02	2	1106001800-2	7.94091608744355e-11
0.7	Q02	3	1106004310-15	5.85219384010464e-11
0.7	Q03	1	1106000874-15	7.36930136918297e-10
0.7	Q03	2	1106007104-11	7.36930136918297e-10
0.7	Q03	3	1106000874-7	6.69239066228927e-10
0.7	Q04	1	1106053470-13	3.47229193930997e-10
0.7	Q04	2	1106004310-13	3.19862935635821e-10
0.7	Q04	3	1106004310-19	9.47598446402929e-11
0.7	Q05	1	1106012003-5	2.67263561232214e-05
0.7	Q05	2	1106005673-17	1.07040855462854e-05
0.7	Q05	3	1106005673-16	8.18455645548192e-06
0.7	Q06	1	1106023032-13	1.44346351103603e-11
0.7	Q06	2	1106053552-17	3.64880410933301e-12
0.7	Q06	3	1106053552-12	1.7805310915594e-12
0.8	Q01	1	1106023032-13	2.36150852859401e-11
0.8	Q01	2	1106053552-17	3.86017489605195e-12
0.8	Q01	3	1106053552-12	1.84982901710395e-12
0.8	Q02	1	1106004310-14	1.83838099968793e-10
0.8	Q02	2	1106001800-2	1.23059273815733e-10
0.8	Q02	3	1106004310-15	5.9561184218432e-11

0.8	Q03	1	1106000874-7	7.90404389773715e-10
0.8	Q03	2	1106007104-11	6.38767078591801e-10
0.8	Q03	3	1106000874-15	6.38767078591801e-10
0.8	Q04	1	1106053470-13	5.57389495978689e-10
0.8	Q04	2	1106004310-13	5.11399590622088e-10
0.8	Q04	3	1106004310-19	1.39870236243435e-10
0.8	Q05	1	1106012003-5	3.85493793587109e-05
0.8	Q05	2	1106005673-17	1.52939915082499e-05
0.8	Q05	3	1106005673-16	1.15929017168169e-05
0.8	Q06	1	1106023032-13	2.36150852859401e-11
0.8	Q06	2	1106053552-17	3.86017489605195e-12
0.8	Q06	3	1106053552-12	1.84982901710395e-12
0.9	Q01	1	1106023032-13	3.68833271898275e-11
0.9	Q01	2	1106053552-17	2.91936306048164e-12
0.9	Q01	3	1106053552-12	1.37832196205894e-12
0.9	Q02	1	1106004310-14	2.84719202873697e-10
0.9	Q02	2	1106001800-2	1.83814202199912e-10
0.9	Q02	3	1106004310-15	4.36355732594735e-11
0.9	Q03	1	1106000874-7	9.18608652397399e-10
0.9	Q03	2	1106053514-7	6.69456942547889e-10
0.9	Q03	3	1106000874-15	4.05370118231744e-10
0.9	Q04	1	1106053470-13	8.58581055636505e-10
0.9	Q04	2	1106004310-13	7.85067433976986e-10
0.9	Q04	3	1106004310-19	2.00087658984115e-10
0.9	Q05	1	1106012003-5	5.34258070367451e-05
0.9	Q05	2	1106005673-17	2.10376351342526e-05

0.9	Q05	3	1106005673-16	1.58345891900559e-05
0.9	Q06	1	1106023032-13	3.68833271898275e-11
0.9	Q06	2	1106053552-17	2.91936306048164e-12
0.9	Q06	3	1106053552-12	1.37832196205894e-12

Relevance Judgment

Setelah mendapatkan hasil top 50 dari keseluruhan model, maka dihitung recall dan precision serta mean average precision dari hasil masing-masing model. Relevansi dilakukan dengan cara mencocokkan dokumen yang teretrieved dengan relevance judgment. Pada program saya hanya melakukan perhitungan precision saat dokumen yang teretrieved adalah relevan.

```
#Relevansi Judgment
open (NEWFILE,"relevant_judgment.txt") or die ("Can't open the file");

while($lines = <NEWFILE>){
      my @array = split /\s+/, $lines;
      $relevanJ{$array[0]}{$array[1]} = $array[2];
}
close(NEWFILE);

sub relevanceJudgment{
      my($namefile,%hash)=@_;
      open (FILE1, ">",$namefile) or die ("can't open file");

      my %Pr;
      my $nquery = keys %query;
      my $average;
      print FILE1"Query,Rank,Document,Score\n";

      foreach my $q (sort keys %hash){
            my $count1 =1;
            my $relevanD= keys %{$relevanJ{$q}};
            my $retrieved=0;
            my $nrelevant=0;
            my $relevant =0;
            my $getdoc;
            my $precision;
            my $recall;
            foreach my $d(sort {$hash{$q}{$b} <=> $hash{$q}{$a}} keys
%{$hash{$q}}){
                  if($count1 <= 50){
                        $retrieved++;
                        if($relevanJ{$q}{$d}){
                              $relevant++;
                              $getdoc = $d;
```

```
                                    $precision =
$relevant/($relevant+$nrelevant);
                                    $recall= $relevant/($relevanD);
                                    $Pr{$q} += $precision;

                              }else{
                                    $nrelevant++;
                              }
                        }
                        $count1++;
                  }
                  $Pr{$q} = $Pr{$q}/$relevant;
                  $average += $Pr{$q};
            }

            $average = ($average/$nquery)*100;
            print"Mean Average precision = $average %\n\n";
}
```

Hasil dari relevance Judgment:

Cossine Similarity dengan TF

Query	Precision	Recall	Average Precision	Document Retrieved	Document Relevant
Q01	0.1	0.227272727272727	0.221464413722478	5	22
Q02	0.34	0.62962962962963	0.327515424274331	17	27
Q03	0.06	0.75	0.0807646356033453	3	4
Q04	0.24	0.75	0.483498058706603	12	16
Q05	0.08	0.307692307692308	0.149220571095571	4	13
Q06	0.1	0.227272727272727	0.218964413722478	5	22

Dengan nilai MAP = 24.6544647655837 %

Cossine Similarity dengan TF-IDF

Query	Precision	Recall	Average Precision	Document Retrieved	Document Relevant
Q01	0.1	0.227272727272727	0.24336917562724	5	22
Q02	0.36	0.666666666666667	0.59993726058158	18	27
Q03	0.06	0.75	0.0718143738977072	3	4
Q04	0.24	0.75	0.564574233779161	12	16
Q05	0.1	0.384615384615385	0.146159154854807	5	13
Q06	0.1	0.227272727272727	0.238240970499035	5	22

Mean Average precision cossine similarity TF-IDF pada = 31.7592387894226 %

Hasil Unigram Model:

Lamda	Query	Precision	Recall	Average Precision	Document Retrieved	Document Relevant
0.3	Q01	0.12	0.272727272727273	0.184268405086074	6	22
0.3	Q02	0.32	0.592592592592593	0.661652618215118	16	27
0.3	Q03	0.06	0.75	0.0767773892773893	3	4
0.3	Q04	0.26	0.8125	0.4018578565891	13	16
0.3	Q05	0.12	0.461538461538462	0.262072122209317	6	13
0.3	Q06	0.12	0.272727272727273	0.186217722824866	6	22
0.4	Q01	0.12	0.272727272727273	0.185285883670377	6	22
0.4	Q02	0.34	0.62962962962963	0.668695316601891	17	27
0.4	Q03	0.06	0.75	0.0941919191919192	3	4
0.4	Q04	0.26	0.8125	0.434644570401093	13	16
0.4	Q05	0.12	0.461538461538462	0.265506840342367	6	13
0.4	Q06	0.12	0.272727272727273	0.187124118964495	6	22
0.5	Q01	0.12	0.272727272727273	0.201475542543486	6	22
0.5	Q02	0.36	0.666666666666667	0.656464118807706	18	27
0.5	Q03	0.06	0.75	0.0941919191919192	3	4
0.5	Q04	0.24	0.75	0.484622956408671	12	16
0.5	Q05	0.12	0.461538461538462	0.271572215120602	6	13
0.5	Q06	0.12	0.272727272727273	0.199637307249368	6	22
0.6	Q01	0.12	0.272727272727273	0.200380028580325	6	22
0.6	Q02	0.38	0.703703703703704	0.631654795750906	19	27
0.6	Q03	0.06	0.75	0.0946969696969697	3	4
0.6	Q04	0.18	0.5625	0.608306614885562	9	16
0.6	Q05	0.12	0.461538461538462	0.276456429904706	6	13
0.6	Q06	0.12	0.272727272727273	0.200380028580325	6	22
0.7	Q01	0.12	0.272727272727273	0.201363379122293	6	22
0.7	Q02	0.38	0.703703703703704	0.625426772693819	19	27
0.7	Q03	0.06	0.75	0.093974358974359	3	4
0.7	Q04	0.18	0.5625	0.673809523809524	9	16
0.7	Q05	0.12	0.461538461538462	0.278632127770059	6	13
0.7	Q06	0.12	0.272727272727273	0.202997366050397	6	22

0.8	Q01	0.12	0.272727272727273	0.202139037433155	6	22
0.8	Q02	0.36	0.666666666666667	0.629163125666394	18	27
0.8	Q03	0.08	1	0.0893246437927289	4	4
0.8	Q04	0.18	0.5625	0.704798904798905	9	16
0.8	Q05	0.12	0.461538461538462	0.284823972323972	6	13
0.8	Q06	0.12	0.272727272727273	0.203977272727273	6	22
0.9	Q01	0.12	0.272727272727273	0.207043650793651	6	22
0.9	Q02	0.36	0.666666666666667	0.624466106723081	18	27
0.9	Q03	0.08	1	0.0916514041514041	4	4
0.9	Q04	0.18	0.5625	0.711669606114051	9	16
0.9	Q05	0.12	0.461538461538462	0.293932918932919	6	13
0.9	Q06	0.12	0.272727272727273	0.205205415499533	6	22

Dengan nilai MAP:

1. Mean Average precision unigram model pada lamda 0.3 = 29.5190471248498 %
2. Mean Average precision unigram model pada lamda 0.4 = 30.6923020738224 %
3. Mean Average precision unigram model pada lamda 0.5 = 31.9226045675945 %
4. Mean Average precision unigram model pada lamda 0.6 = 33.3574980268717 %
5. Mean Average precision unigram model pada lamda 0.7 = 34.4969834124782 %
6. Mean Average precision unigram model pada lamda 0.8 = 35.2127362119436 %
7. Mean Average precision unigram model pada lamda 0.9 = 35.5737725970526 %

Kesimpulan dari pemakaian model:

Berdasarkan hasil mean average precision dari top 50 dokumen, Cossine Similarity dengan nilai TF dikali dengan IDF memiliki hasil yang lebih baik dibanding dengan Cossine Similarity dengan nilai TFnya saja. Hal ini dikarenakan kata yang umum memiliki skor lebih tinggi yang biasanya merupakan kata yang tidak penting, tetapi yang jadi masalah adalah ketika kata yang tidak penting tersebut juga memiliki kata yang tidak penting.

Nilai mean average precision pada unigram model mencapai nilai tertinggi pada lamda = 0.9 dengan nilai precision: 35.57 % pada kasus TOP 50. Jika disimpulkan dari ketiga model tersebut score MAP tertinggi diperoleh oleh Unigram Model dengan Mixture model.

Pada kasus eksperimen saya menggunakan kemungkinan kemunculannya pada dokumen dengan. Namun saya menggunakannya dengan skor jumlah dari seluruh TFIDF suatu kata dibagi dengan term yang ada di dokumen. Rumus yang saya gunakan adalah

$$Skor(q,d) = \sum_{i=1}^{q} \frac{TF*IDF}{TD}$$

Koding:

```
my %cobahasil;
foreach my $q(sort keys %query){
	print File "$q\n";
	foreach my $d(sort @doc){
		if($TerminDocument{$d} != 0){
			#hanya menggunakan probabilitas
			my $temp =1;
			foreach my $w(sort keys %{$query{$q}}){
				my $peluang = $IDFofDoc{$w}{$d}/$TerminDocument{$d};
				$temp += $peluang;
			}
			$cobahasil{$q}{$d} = $temp;
		}
	}
}
```

Hasil Relevance Judgment:

Query	Precision	Recall	Average Precision	Document Retrieved	Document Relevant
Q01	0.12	0.272727272727273	0.189714399273223	6	22
Q02	0.34	0.62962962962963	0.418363592008643	17	27
Q03	0.06	0.75	0.118908382066277	3	4
Q04	0.24	0.75	0.358216097958426	12	16
Q05	0.08	0.307692307692308	0.135678210678211	4	13
Q06	0.12	0.272727272727273	0.189714399273223	6	22

Dengan Mean Average precision pada = 23.5099180209667 %

Hasil tidak lebih baik daripada ketiga model, namun mendekati cossine similarity dengan TF.