

* Compare TF-IDF and tobagofwords
* Explain
* Show work



Task

TF-IDF Output TobagofWords

Enter a term: win

.: 35.7803

,: 33.6471

the: 32.8694

to: 25.2341

a: 23.3677

Enter a term: apple

,: 4.8346

.: 3.8144

the: 3.4933

Sutton: 3.2886

pies: 3.2138

Enter a term: student

,: 20.2657

the: 20.1252

.: 18.4400

of: 14.4104

to: 13.9769

Enter a term: window

.: 12.6961

,: 12.2819

the: 9.9853

a: 8.1166

to: 6.7849

Enter a term: Allen

.: 6.4153

the: 5.9481

Dallas: 5.9395

Mavericks: 5.7875

Celtics: 5.4069

Enter a term: water

.: 29.9840

,: 29.1311

the: 26.5292

to: 19.5725

a: 19.2226

Enter a term: win

": 32.0412

game: 30.4851

team: 26.2502

': 22.2146

said: 21.9271

Enter a term: apple

Hubig: 20.5884

pies: 20.5810

Sutton: 19.6744

GBS.: 17.3596

Syndrome: 16.7717

Enter a term: student

fees: 36.8002

subsidy: 31.7031

students: 29.5082

aid: 27.3002

education: 24.7325

Enter a term: window

lawn: 15.4623

UNINTELLIGIBLE: 14.0791

IRWIN: 13.7635

Shipman: 13.5384

I`ve: 13.1989

Enter a term: Enter a term: Allen

Mavericks: 34.9807

Celtics: 32.2026

Dallas: 31.2373

Stackhouse: 29.2649

Terry: 19.9646

Enter a term: Enter a term: water

mist: 31.6681

humidifier: 24.7621

HWM-2030: 21.5931

sump: 18.9232

cool: 18.6596

Enter a term: Enter a term: ­­­

To bagofwords relies more on term frequency and that’s why we get more garbage results. Stop words or TF-IDF improves this greatly. TF-IDF uses the inverse document frequency to get rid of words that appear too many times across all documents.

**public** LatentSemanticAnalysis(List<List<String>> documents, **int** k)

{

vs\_model = **new** VectorSpaceModel();

//List<Term[]> list = vs\_model.toTFIDFs(documents);

List<Term[]> list = **new** ArrayList<>();

**for** (List<String> document : documents)

list.add(vs\_model.toBagOfWords(document, **true**));

**int** T = vs\_model.getTermSize(), D = list.size();

td\_matrix = **new** Basic2DMatrix(T, D);

**for** (**int** docID=0; docID<D; docID++)

**for** (Term term : list.get(docID))

td\_matrix.set(term.getID(), docID, term.getScore());

////toLSA(k);

}

**public** List<ObjectDoublePair<String>> getTopSimilarTerms(String term, **int** k)

{

List<ObjectDoublePair<String>> list = **new** ArrayList<>();

**int** termID= vs\_model.getID(term);

**if**(termID < 0) **return** list;

**for**(**int** i = td\_matrix.rows()-1; i>=0;i--)

{

**if**((i)!=termID)

list.add(**new** ObjectDoublePair<String> ( vs\_model.getTerm(i),

getCosineSimilarityTerm(termID,i)));

}

Collections.*sort*(list, Collections.*reverseOrder*());

**return** (k> list.size())? list : list.subList(0, k);

}