

ClusteringSolutions

May 26, 2017

1 Solutions to the Clustering Task

1.1 Clustering

The main clustering package for julia, is unexpectedly, named [Clustering.jl](#) - It supports K-means, K-medoids, Affinity Propagation, DBSCAN - It also supports hierarchical clustering, but that is not currently in the docs.

You'll also want [Distances.jl](#) for all your distance metric needs. It is traditional with word2vec to use cosine distance.

1.1.1 Affinity Propagation

If you set the availability right, it can get a breakdown where the ball-sports and clustered separately from the other sports. Though you may have problems with some of the cities being classes as sports, as this word2vec representation was trained on a dump of wikipedia taken in 2014, and there are a lot of sports pages talking about the Athens and Beijing olympics.

2 First we load up some data

For the the example presented here, we will use a subset of Word Embedding, trained using [Word2Vec.jl](#). These are 100 dimensional vectors, which encode syntactic and semantic information about words.

```
In [1]: using JLD
         embeddings = load("../assets/ClusteringAndDimensionalityReduction.jld", "em
```

```
Out[1]: Dict{String,Array{Float32,1}} with 185 entries:
  "ferret"      => Float32[0.0945707,-0.435267,0.0109875,-0.107674,0.16900
  "gymnastics"  => Float32[-0.269173,-0.343412,-0.00603042,-0.186179,0.034
  "vegas"       => Float32[-0.00530534,-0.264874,0.0167432,-0.289836,-0.14
  "archery"     => Float32[0.0279714,-0.485648,0.105468,-0.0696941,0.18280
  "jacksonville" => Float32[-0.418758,-0.0284594,0.00847164,-0.0989162,0.09
  "ankara"      => Float32[-0.139109,0.0872892,0.749557,-0.0308427,-0.0936
  "pentathlon"  => Float32[-0.357405,-0.379595,-0.134314,-0.31008,-0.02458
  "seoul"       => Float32[0.0274904,-0.153844,-0.0936614,-0.0269344,-0.09
  "china"       => Float32[0.132423,-0.515862,-0.0381339,-0.287565,-0.2852
  "korea"       => Float32[0.236904,-0.128355,-0.0816942,-0.0702621,-0.148
```

```

"argentina"      => Float32[-0.113967,-0.437523,-0.226014,-0.439572,-0.2300
"mozambique"     => Float32[0.309411,-0.13457,-0.632055,-0.309943,0.040591,
"iraq"           => Float32[-0.260673,0.0356129,0.104878,0.103836,-0.17918,
"baku"           => Float32[0.182572,0.156322,0.225807,-0.0933851,-0.246997
"jakarta"        => Float32[0.13052,-0.408592,-0.0138496,-0.415052,0.21523,
"bogotá"         => Float32[-0.26368,-0.292844,-0.338501,-0.278793,-0.06909
"sacramento"     => Float32[-0.217914,-0.116757,-0.213111,-0.13627,-0.02413
"dhaka"          => Float32[-0.0264262,-0.256298,0.0922423,-0.711511,-0.329
"kyiv"           => Float32[-0.0527193,0.219892,-0.298013,-0.594799,-0.4527
"houston"        => Float32[-0.301442,-0.133911,-0.17504,-0.0391225,-0.0525
"italy"          => Float32[0.246153,-0.0510639,-0.143408,-0.149572,-0.2291
"francisco"      => Float32[-0.342338,-0.200734,-0.347174,-0.228947,-0.1255
"baghdad"        => Float32[-0.309163,0.00524779,0.287937,0.0294381,-0.1730
"dog"            => Float32[0.0509182,-0.479764,0.0209584,-0.0409415,0.0650
"kabul"          => Float32[-0.0282727,-0.108688,0.249284,0.119064,-0.16364
:                => :

```

```

In [2]: all_words = collect(keys(embeddings))
        display(all_words)
        embeddings_mat = hcat(getindex.([embeddings], all_words)...)

```

```

185-element Array{String,1}:

```

```

"ferret"
"gymnastics"
"vegas"
"archery"
"jacksonville"
"ankara"
"pentathlon"
"seoul"
"china"
"korea"
"argentina"
"mozambique"
"iraq"
:
"volleyball"
"luanda"
"ghana"
"warsaw"
"accra"
"indianapolis"
"las"
"russia"
"columbus"
"thailand"
"mesa"
"goose"

```

```

Out[2]: 100×185 Array{Float32,2}:
  0.0945707 -0.269173 ... 0.0859109 -0.215521 0.118283
 -0.435267 -0.343412 -0.185847 -0.0846722 -0.40088
  0.0109875 -0.00603042 0.131935 -0.452262 0.0091058
 -0.107674 -0.186179 -0.221565 -0.115309 0.0121521
  0.169001 0.0342606 -0.0558827 -0.373113 -0.0509757
 -0.0564122 -0.137685 ... -0.0252548 -0.264813 -0.24657
 -0.249841 -0.162321 0.0430546 0.0958876 0.0397347
 -0.115424 -0.253833 0.161854 -0.274667 -0.120246
 -0.302291 0.0844513 -0.263644 -0.158253 -0.0829336
 -0.0232056 0.138056 -0.476437 -0.1159 0.0935187
 -0.0826832 0.0510365 ... -0.190116 -0.00022561 -0.338357
 -0.11338 0.0767575 -0.0493041 -0.252975 -0.0785137
 -0.255015 -0.591677 0.0772709 0.180385 -0.134259
  ⋮
 -0.191331 -0.290943 -0.164481 -0.0834075 0.182234
  0.0188895 -0.594902 -0.365009 0.0104588 0.205071
 -0.1854 0.169166 ... -0.175704 -0.00672958 -0.047146
 -0.0505824 0.251584 -0.468982 0.257763 -0.0606379
 -0.169504 0.189358 -0.0463411 -0.0674982 -0.222475
  0.00213797 -0.113193 0.0145402 -0.192454 -0.087933
 -0.0281012 -0.07121 0.158821 -0.00851574 -0.51206
 -0.064161 -0.132502 ... -0.439959 -0.512601 -0.186174
 -0.161479 -0.0524493 0.170235 -0.123051 0.0789853
  0.0849455 -0.178514 -0.0547684 -0.578576 0.154197
  0.0256315 -0.154586 0.0279115 -0.28596 0.0913349
 -0.0626861 0.312328 0.338198 0.0463355 -0.14829

```

```

In [3]: using Clustering
        using Distances

```

```

similarity = 1f0 - pairwise(CosineDist(), embeddings_mat)
availability = 0.01*ones(size(similarity,1))
# tweaking availability is how you control number of clusters
# it is the diagonal of the similarity matrix
similarity[diagind(size(similarity)...)] = availability
aprop = affinityprop(similarity)

```

```

Out[3]: Clustering.AffinityPropResult([10,17,24,49,77,84,87,88,91,107,136,148,161,1

```

```

In [4]: for (cluster_ii, exemplar_ind) in enumerate(aprop.exemplars)
        println("-"^32)
        println("Exemplar: ", all_words[exemplar_ind])
        cluster_member_inds = find(assignments(aprop).==cluster_ii)
        println(join(getindex.([all_words], cluster_member_inds), ", "))
    end

```

Exemplar: korea

seoul, china, korea, pyongyang, japan, vietnam, tokyo, hanoi, taipei

Exemplar: sacramento

vegas, sacramento, francisco, tucson, seattle, san, albuquerque, denver, portland,

Exemplar: dog

ferret, dog, goldfish, cattle, dove, yak, duck, llama, mouse, alpaca, pigeon, guinea

Exemplar: indonesia

jakarta, bangkok, myanmar, indonesia, manila, malaysia, philippines, singapore, thailand

Exemplar: iran

iraq, kabul, uzbekistan, tehran, iran, yemen, afghanistan

Exemplar: cairo

ankara, baghdad, algiers, khartoum, rabat, beirut, cairo, algeria, morocco, damascus

Exemplar: vienna

italy, rome, berlin, vienna, stockholm, budapest, paris

Exemplar: moscow

baku, kyiv, bucharest, tashkent, moscow, minsk, ukraine, russia

Exemplar: colombia

argentina, bogotá, lima, madrid, venezuela, havana, peru, colombia, brazil, brasil

Exemplar: wales

australia, england, london, wales, britain, rugby, ireland

Exemplar: poland

south, france, germany, poland, warsaw

Exemplar: weightlifting

gymnastics, archery, pentathlon, shooting, diving, swimming, fencing, rowing, taekwondo

Exemplar: bangladesh

dhaka, nepal, pakistan, india, bangladesh

Exemplar: uganda

mozambique, madagascar, congo, yaoundé, pretoria, sudan, tanzania, angola, nigeria,

Exemplar: volleyball

basketball, handball, polo, tennis, football, golf, hockey, soccer, badminton, volleyball

Exemplar: columbia

jacksonville, houston, nashville, raleigh, phoenix, washington, detroit, milwaukee,
