Vivek sheth 0890993

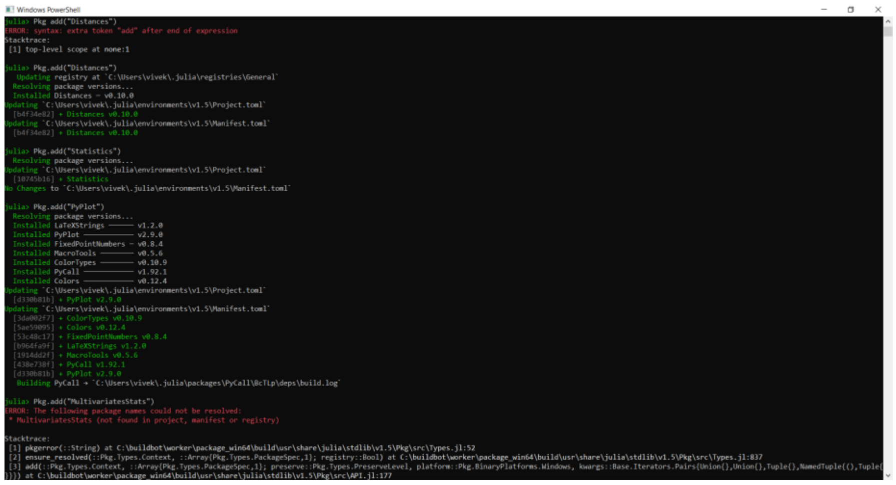
Artificial Intelligence project 2:

Development and Experimentation with Reasoning with Word Vectors in Julia (Word Embedding Utilities)

Following are the steps that I have used to do the word embedding:

- I have used the GloVe (global vectors for word representation) and I have also used our professor's L15 slides as a reference.
- First, I added the following directories in Julia:

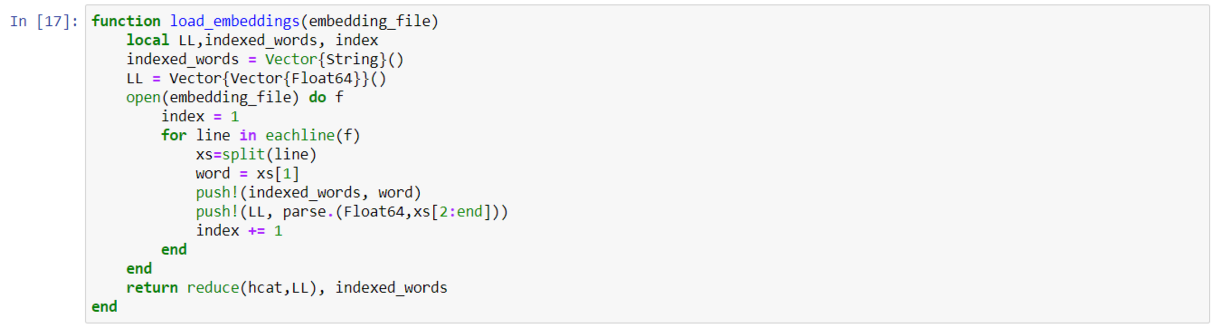
  Distances
  Statistics
  MultivariateStats
  PyPlot
  WordTokenizers
  TextAnalysis
  DelimitedFiles



```
julia> Pkg add("Distances")
ERROR: syntax: extra token "add" after end of expression
Stacktrace:
 [1] top-level scope at none:1

julia> Pkg.add("Distances")
   Updating registry at `C:\Users\vivek\.julia\registries\General`
  Resolving package versions...
  Installed Distances ─ v0.10.0
Updating `C:\Users\vivek\.julia\environments\v1.5\Project.toml`
  [b4f34e82] + Distances v0.10.0
Updating `C:\Users\vivek\.julia\environments\v1.5\Manifest.toml`
  [b4f34e82] + Distances v0.10.0

julia> Pkg.add("Statistics")
  Resolving package versions...
Updating `C:\Users\vivek\.julia\environments\v1.5\Project.toml`
  [10745b16] + Statistics
No Changes to `C:\Users\vivek\.julia\environments\v1.5\Manifest.toml`

julia> Pkg.add("PyPlot")
  Resolving package versions...
  Installed LaTeXStrings ────── v1.2.0
  Installed PyPlot ──────────── v2.9.0
  Installed FixedPointNumbers ─ v0.8.4
  Installed MacroTools ──────── v0.5.6
  Installed ColorTypes ──────── v0.10.9
  Installed PyCall ──────────── v1.92.1
  Installed Colors ──────────── v0.12.4
Updating `C:\Users\vivek\.julia\environments\v1.5\Project.toml`
  [d330b81b] + PyPlot v2.9.0
Updating `C:\Users\vivek\.julia\environments\v1.5\Manifest.toml`
  [3da002f7] + ColorTypes v0.10.9
  [5ae59095] + Colors v0.12.4
  [53c48c17] + FixedPointNumbers v0.8.4
  [b964fa9f] + LaTeXStrings v1.2.0
  [1914dd2f] + MacroTools v0.5.6
  [438e738f] + PyCall v1.92.1
  [d330b81b] + PyPlot v2.9.0
   Building PyCall → `C:\Users\vivek\.julia\packages\PyCall\BcTLp\deps\build.log`

julia> Pkg.add("MultivariatesStats")
ERROR: The following package names could not be resolved:
 * MultivariatesStats (not found in project, manifest or registry)

Stacktrace:
 [1] pkgerror(::String) at C:\buildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.5\Pkg\src\Types.jl:52
 [2] ensure_resolved(::Pkg.Types.Context, ::Array{Pkg.Types.PackageSpec,1}; registry::Bool) at C:\buildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.5\Pkg\src\Types.jl:837
 [3] add(::Pkg.Types.Context, ::Array{Pkg.Types.PackageSpec,1}; preserve::Pkg.Types.PreserveLevel, platform::Pkg.BinaryPlatforms.Windows, kwargs::Base.Iterators.Pairs{Union{},Union{},Tuple{},NamedTuple{(),Tuple{
}}}) at C:\buildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.5\Pkg\src\API.jl:177
```

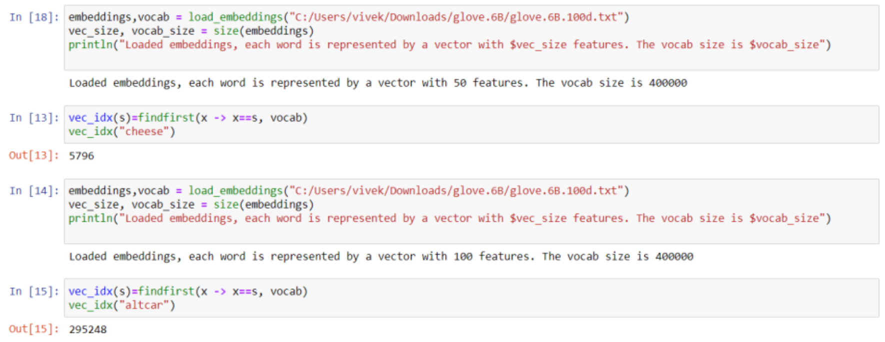
In [1]:
```julia
using Distances, Statistics
using MultivariateStats
using PyPlot
using WordTokenizers
using TextAnalysis
using DelimitedFiles
```

- After that I started calling the Load_embeddings fuction in which all the embeddings are added from the text file of the Glove.



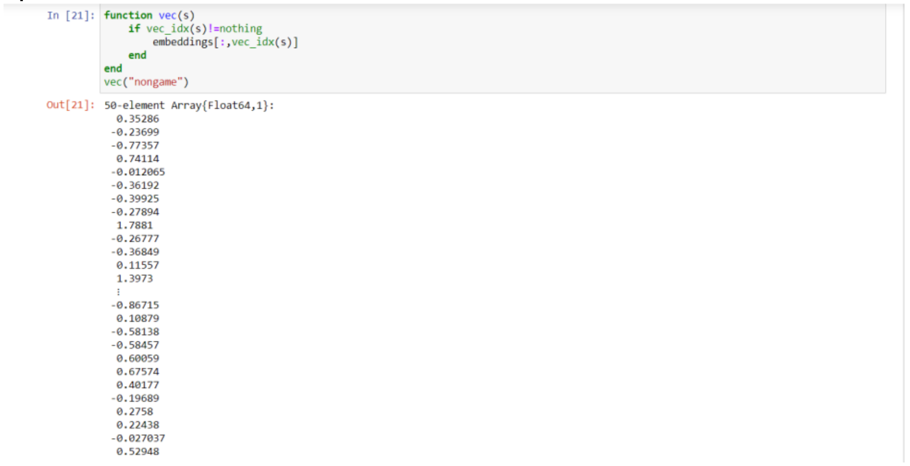
```
In [17]: function load_embeddings(embedding_file)
             local LL,indexed_words, index
             indexed_words = Vector{String}()
             LL = Vector{Vector{Float64}}()
             open(embedding_file) do f
                 index = 1
                 for line in eachline(f)
                     xs=split(line)
                     word = xs[1]
                     push!(indexed_words, word)
                     push!(LL, parse.(Float64,xs[2:end]))
                     index += 1
                 end
             end
             return reduce(hcat,LL), indexed_words
         end
```

- Load_embedding function displays the vocab size and also if we find any specific word it will display it's position.



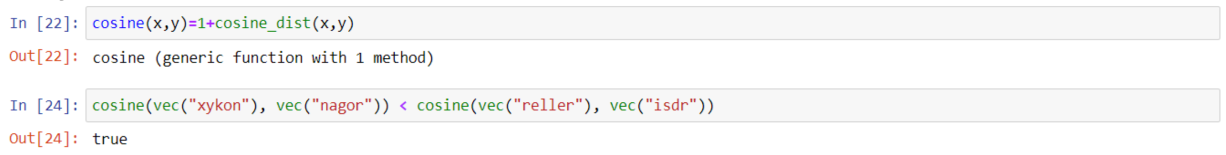
```
In [18]: embeddings,vocab = load_embeddings("C:/Users/vivek/Downloads/glove.6B/glove.6B.100d.txt")
         vec_size, vocab_size = size(embeddings)
         println("Loaded embeddings, each word is represented by a vector with $vec_size features. The vocab size is $vocab_size")

         Loaded embeddings, each word is represented by a vector with 50 features. The vocab size is 400000

In [13]: vec_idx(s)=findfirst(x -> x==s, vocab)
         vec_idx("cheese")

Out[13]: 5796

In [14]: embeddings,vocab = load_embeddings("C:/Users/vivek/Downloads/glove.6B/glove.6B.100d.txt")
         vec_size, vocab_size = size(embeddings)
         println("Loaded embeddings, each word is represented by a vector with $vec_size features. The vocab size is $vocab_size")

         Loaded embeddings, each word is represented by a vector with 100 features. The vocab size is 400000

In [15]: vec_idx(s)=findfirst(x -> x==s, vocab)
         vec_idx("altcar")

Out[15]: 295248
```

- Then I have declared thr Vec() function. In which it shows the 50 element array of any specific word.



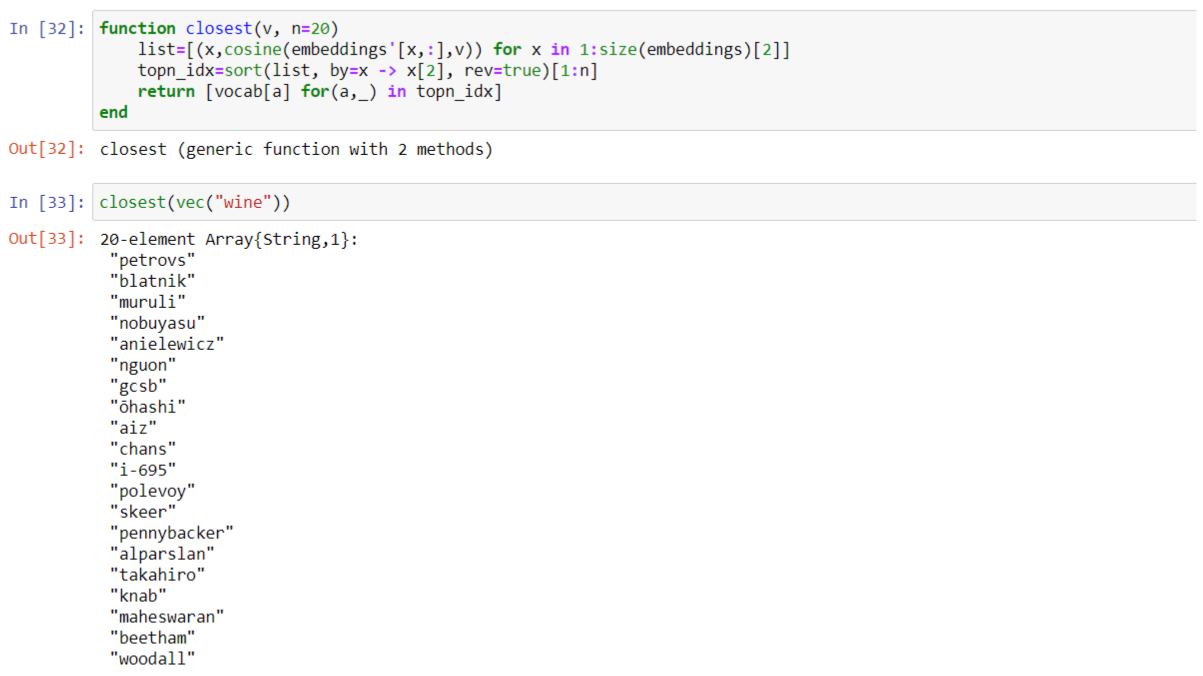
```
In [21]: function vec(s)
             if vec_idx(s)!=nothing
                 embeddings[:,vec_idx(s)]
             end
         end
         vec("nongame")

Out[21]: 50-element Array{Float64,1}:
           0.35286
          -0.23699
          -0.77357
           0.74114
          -0.012065
          -0.36192
          -0.39925
          -0.27894
           1.7881
          -0.26777
          -0.36849
           0.11557
           1.3973
           ⋮
          -0.86715
           0.10879
          -0.58138
          -0.58457
           0.60059
           0.67574
           0.40177
          -0.19689
           0.2758
           0.22438
          -0.027037
           0.52948
```

- It also calls the cosine function which shows the greater or the smaller word size by calling true or false.



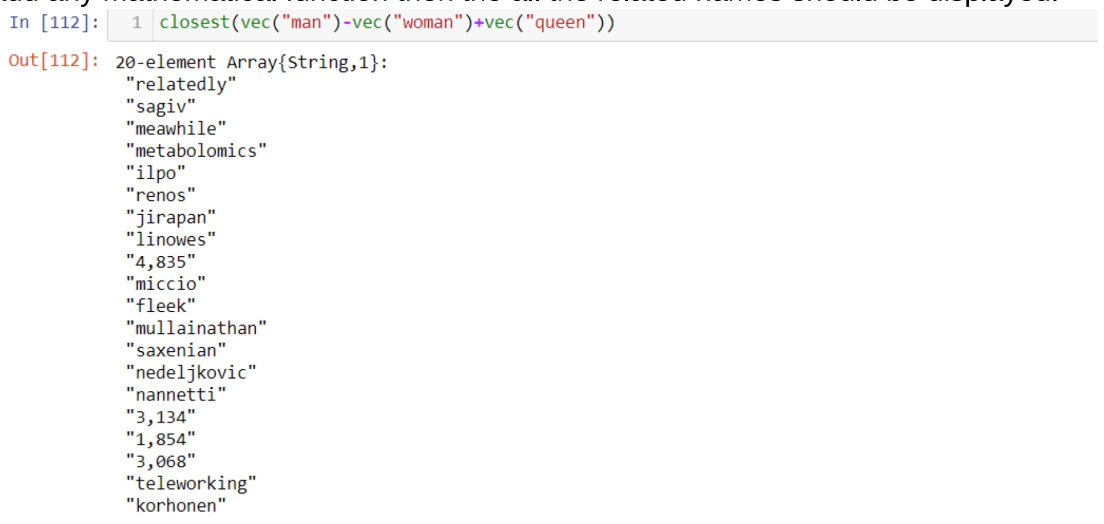
```
In [22]: cosine(x,y)=1+cosine_dist(x,y)

Out[22]: cosine (generic function with 1 method)

In [24]: cosine(vec("xykon"), vec("nagor")) < cosine(vec("reller"), vec("isdr"))

Out[24]: true
```

- The next function I've used is the closest in which if we call any single word then the 20 element array will be displayed.



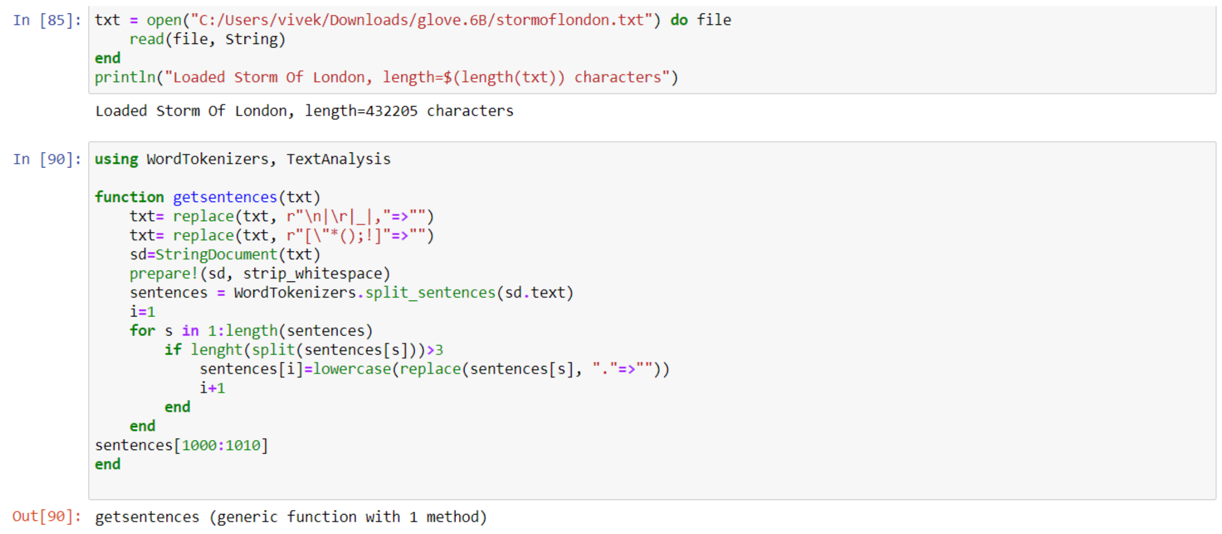
```
In [32]: function closest(v, n=20)
             list=[(x,cosine(embeddings'[x,:],v)) for x in 1:size(embeddings)[2]]
             topn_idx=sort(list, by=x -> x[2], rev=true)[1:n]
             return [vocab[a] for(a,_) in topn_idx]
         end

Out[32]: closest (generic function with 2 methods)

In [33]: closest(vec("wine"))

Out[33]: 20-element Array{String,1}:
          "petrovs"
          "blatnik"
          "muruli"
          "nobuyasu"
          "anielewicz"
          "nguon"
          "gcsb"
          "ōhashi"
          "aiz"
          "chans"
          "i-695"
          "polevoy"
          "skeer"
          "pennybacker"
          "alparslan"
          "takahiro"
          "knab"
          "maheswaran"
          "beetham"
          "woodall"
```

- If we call the closest function with a vec sysntax as well as any specific words in that and add any mathematical function then the all the related names should be displayed.



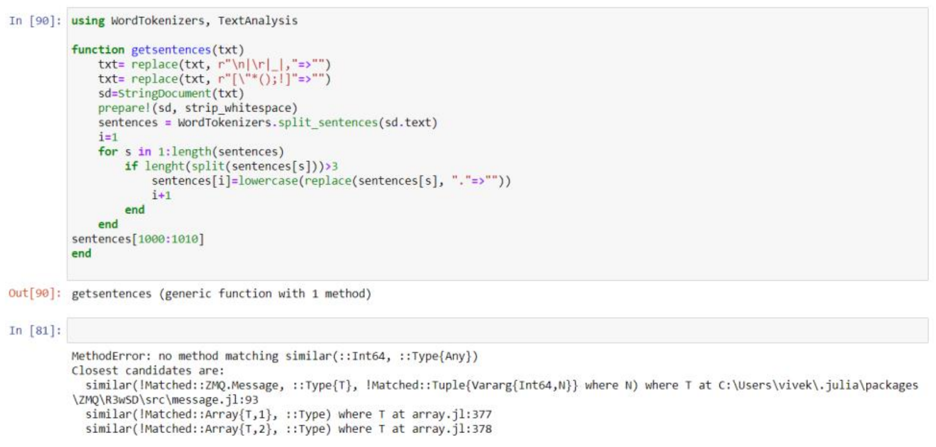
```
In [112]: closest(vec("man")-vec("woman")+vec("queen"))

Out[112]: 20-element Array{String,1}:
           "relatedly"
           "sagiv"
           "meawhile"
           "metabolomics"
           "ilpo"
           "renos"
           "jirapan"
           "linowes"
           "4,835"
           "miccio"
           "fleek"
           "mullainathan"
           "saxenian"
           "nedeljkovic"
           "nannetti"
           "3,134"
           "1,854"
           "3,068"
           "teleworking"
           "korhonen"
```

- I have also tried the sentence embedding but it was not fully running, I had some doubts.



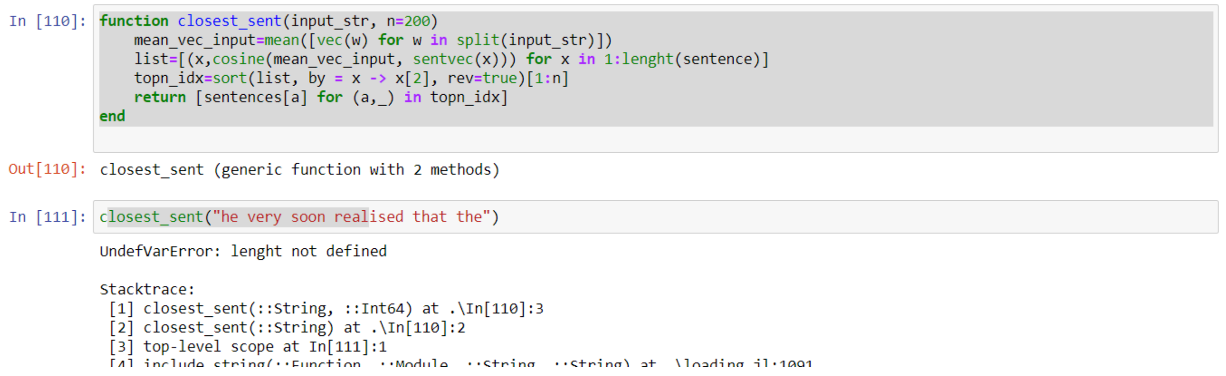
```
In [85]: txt = open("C:/Users/vivek/Downloads/glove.6B/stormoflondon.txt") do file
             read(file, String)
         end
         println("Loaded Storm Of London, length=$(length(txt)) characters")

         Loaded Storm Of London, length=432205 characters

In [90]: using WordTokenizers, TextAnalysis

         function getsentences(txt)
             txt= replace(txt, r"\n|\r|_|,"=>"")
             txt= replace(txt, r"[\"*();!]"=>"")
             sd=StringDocument(txt)
             prepare!(sd, strip_whitespace)
             sentences = WordTokenizers.split_sentences(sd.text)
             i=1
             for s in 1:length(sentences)
                 if lenght(split(sentences[s]))>3
                     sentences[i]=lowercase(replace(sentences[s], "."=>""))
                     i+1
                 end
             end
         sentences[1000:1010]
         end

Out[90]: getsentences (generic function with 1 method)
```

- In that I have used the book “The storm of London”. I loaded the book and it shows us the book uth total numbers of charcters in it.
- I tried making the closestsent() function but it was showing some errors.



```
In [90]: using WordTokenizers, TextAnalysis

         function getsentences(txt)
             txt= replace(txt, r"\n|\r|_|,"=>"")
             txt= replace(txt, r"[\"*();!]"=>"")
             sd=StringDocument(txt)
             prepare!(sd, strip_whitespace)
             sentences = WordTokenizers.split_sentences(sd.text)
             i=1
             for s in 1:length(sentences)
                 if lenght(split(sentences[s]))>3
                     sentences[i]=lowercase(replace(sentences[s], "."=>""))
                     i+1
                 end
             end
         sentences[1000:1010]
         end

Out[90]: getsentences (generic function with 1 method)
```

```
In [81]:

         MethodError: no method matching similar(::Int64, ::Type{Any})
         Closest candidates are:
           similar(!Matched::ZMQ.Message, ::Type{T}, !Matched::Tuple{Vararg{Int64,N}} where N) where T at C:\Users\vivek\.julia\packages
         \ZMQ\R3wSD\src\message.jl:93
           similar(!Matched::Array{T,1}, ::Type) where T at array.jl:377
           similar(!Matched::Array{T,2}, ::Type) where T at array.jl:378
```



```
In [110]: function closest_sent(input_str, n=200)
              mean_vec_input=mean([vec(w) for w in split(input_str)])
              list=[(x,cosine(mean_vec_input, sentvec(x))) for x in 1:lenght(sentence)]
              topn_idx=sort(list, by = x -> x[2], rev=true)[1:n]
              return [sentences[a] for (a,_) in topn_idx]
          end

Out[110]: closest_sent (generic function with 2 methods)

In [111]: closest_sent("he very soon realised that the")

          UndefVarError: lenght not defined

          Stacktrace:
           [1] closest_sent(::String, ::Int64) at .\In[110]:3
           [2] closest_sent(::String) at .\In[110]:2
           [3] top-level scope at In[111]:1
           [4] include_string(::Function, ::Module, ::String, ::String) at .\loading.jl:1091
```