```
# Installation cell
%%shell
if ! command -v julia 3>&1 > /dev/null
then
    wget 'https://julialang-s3.julialang.org/bin/linux/x64/1.5/julia-1.5.0-linux-x86_64.tar.g
        -O /tmp/julia.tar.gz
    tar -x -f /tmp/julia.tar.gz -C /usr/local --strip-components 1
    rm /tmp/julia.tar.gz
fi
julia -e 'using Pkg; pkg"add IJulia; precompile;"'
echo 'Done'
```

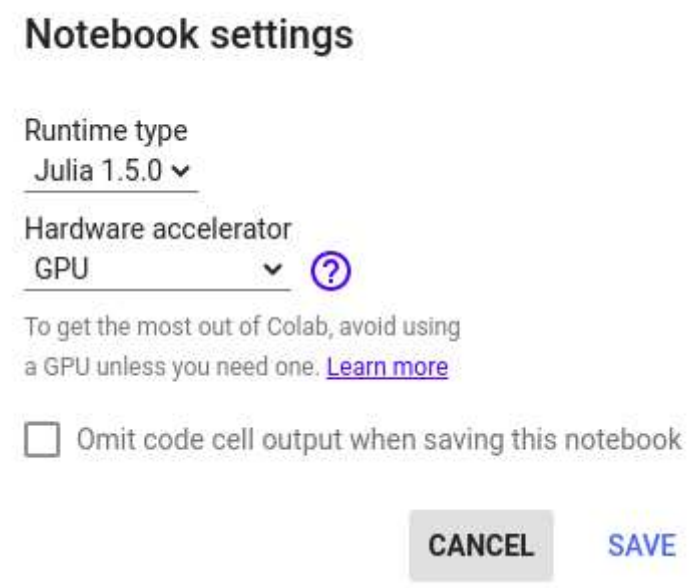
```
           Added registry `General` to `~/.julia/registries/General`
      Resolving package versions...
       Installed ZeroMQ_jll ──────── v4.3.2+5
       Installed Conda ───────────── v1.5.0
       Installed VersionParsing ──── v1.2.0
       Installed SoftGlobalScope ─── v1.1.0
       Installed ZMQ ─────────────── v1.2.1
       Installed JLLWrappers ─────── v1.1.3
       Installed IJulia ──────────── v1.23.1
       Installed JSON ────────────── v0.21.1
       Installed Artifacts ───────── v1.3.0
       Installed MbedTLS_jll ─────── v2.16.8+1
       Installed MbedTLS ─────────── v1.0.3
       Installed Parsers ─────────── v1.0.12
     Downloading artifact: ZeroMQ
     ######################################################################## 100.0%
     Downloading artifact: MbedTLS
     ######################################################################## 100.0%
     Updating `~/.julia/environments/v1.5/Project.toml`
       [7073ff75] + IJulia v1.23.1
     Updating `~/.julia/environments/v1.5/Manifest.toml`
       [56f22d72] + Artifacts v1.3.0
       [8f4d0f93] + Conda v1.5.0
       [7073ff75] + IJulia v1.23.1

       [692b3bcd] + JLLWrappers v1.1.3
       [682c06a0] + JSON v0.21.1
       [739be429] + MbedTLS v1.0.3
       [c8ffd9c3] + MbedTLS_jll v2.16.8+1
       [69de0a69] + Parsers v1.0.12
       [b85f4697] + SoftGlobalScope v1.1.0
       [81def892] + VersionParsing v1.2.0
       [c2297ded] + ZMQ v1.2.1
       [8f1865be] + ZeroMQ_jll v4.3.2+5
       [2a0f44e3] + Base64
       [ade2ca70] + Dates
       [8ba89e20] + Distributed
       [7b1f6079] + FileWatching
       [b77e0a4c] + InteractiveUtils
       [76f85450] + LibGit2
       [8f399da3] + Libdl
       [56ddb016] + Logging
       [d6f4376e] + Markdown
```

```
[a63ad114] + Mmap
[44cfe95a] + Pkg
[de0858da] + Printf
[3fa0cd96] + REPL
[9a3f8284] + Random
[ea8e919c] + SHA
[9e88b42a] + Serialization
[6462fe0b] + Sockets
[8dfed614] + Test
[cf7118a7] + UUIDs
[4ec0a83e] + Unicode
  Building Conda → `~/.julia/packages/Conda/x5ml4/deps/build.log`
  Building IJulia → `~/.julia/packages/IJulia/IDNmS/deps/build.log`
Precompiling project...
Done
```

After you run the first cell (the the cell directly above this text), go to Colab's menu bar and select **Edit** and select **Notebook settings** from the drop down. Select *Julia 1.5.0* as the runtime and *GPU* as the hadware accelerator.

You should see somthing like this:



Click on SAVE

We are ready to get going

```
VERSION
```

```
v"1.5.0"
```

```
import Pkg
```

```
Pkg.add("Word2Vec")
```

```
   Updating registry at `~/.julia/registries/General`
  Resolving package versions...
  Installed Word2Vec_jll ─ v0.1.0+0
  Installed Word2Vec ──── v0.5.3
######################################################################## 100.0%
Updating `~/.julia/environments/v1.5/Project.toml`
  [c64b6f0f] + Word2Vec v0.5.3
Updating `~/.julia/environments/v1.5/Manifest.toml`
  [c64b6f0f] + Word2Vec v0.5.3
  [9fbe4022] + Word2Vec_jll v0.1.0+0
  [37e2e46d] + LinearAlgebra
  [2f01184e] + SparseArrays
  [10745b16] + Statistics
```

```
Pkg.add("Gadfly")
```

```
  [621f4979] + AbstractFFTs v0.5.0
  [79e6a3ab] + Adapt v2.3.0
  [13072b0f] + AxisAlgorithms v1.0.0
  [324d7699] + CategoricalArrays v0.8.3
  [3da002f7] + ColorTypes v0.10.9
  [5ae59095] + Colors v0.12.4
  [34da2185] + Compat v3.23.0
  [e66e0078] + CompilerSupportLibraries_jll v0.3.4+0
  [a81c6b42] + Compose v0.9.1
  [d38c429a] + Contour v0.5.6
  [7ad07ef1] + CoupledFields v0.2.0
  [9a962f9c] + DataAPI v1.4.0
  [864edb3b] + DataStructures v0.18.8
  [b4f34e82] + Distances v0.10.0
  [31c24e10] + Distributions v0.23.12
  [ffbed154] + DocStringExtensions v0.8.3
  [7a1cc6ca] + FFTW v1.2.4
  [f5851436] + FFTW_jll v3.3.9+6
  [1a297f60] + FillArrays v0.9.7
  [53c48c17] + FixedPointNumbers v0.8.4
  [c91e804a] + Gadfly v1.3.1
  [42e2da0e] + Grisu v1.0.0
  [a1b4810d] + Hexagons v0.2.0
  [9b13fd28] + IndirectArrays v0.5.1
  [1d5cc7b8] + IntelOpenMP_jll v2018.0.3+0
  [a98d9a8b] + Interpolations v0.13.1
  [c8e1da08] + IterTools v1.3.0
  [e5e0dc1b] + Juno v0.8.4
  [5ab0869b] + KernelDensity v0.6.2
  [4345ca2d] + Loess v0.5.2
  [856f044c] + MKL_jll v2020.2.254+0
  [1914dd2f] + MacroTools v0.5.6
  [442fdcdd] + Measures v0.3.1
  [e89f7d12] + Media v0.5.0
  [e1d29d7a] + Missings v0.4.4
  [6fe1bfb0] + OffsetArrays v1.4.0
  [efe28fd5] + OpenSpecFun_jll v0.5.3+4
  [bac558e1] + OrderedCollections v1.3.2
```

```
 [90014a1f] + PDMats v0.10.1
 [1fd47b50] + QuadGK v2.4.1
 [c84ed2f1] + Ratios v0.4.0
 [189a3867] + Reexport v0.2.0
 [ae029012] + Requires v1.1.1
 [79098fc4] + Rmath v0.6.1
 [f50d1b31] + Rmath_jll v0.2.2+1
 [992d4aef] + Showoff v0.3.2
 [a2af1166] + SortingAlgorithms v0.3.1
 [276daf66] + SpecialFunctions v0.10.3
 [90137ffa] + StaticArrays v0.12.5
 [2913bbd2] + StatsBase v0.33.2
 [4c63d2b9] + StatsFuns v0.9.6
 [856f2bd8] + StructTypes v1.1.0
 [efce3f68] + WoodburyMatrices v0.5.3
 [8bb1440f] + DelimitedFiles
 [9fa8497b] + Future
 [9abbd945] + Profile
 [1a1011a3] + SharedArrays
 [4607b0f0] + SuiteSparse
  Building FFTW → `~/.julia/packages/FFTW/DMUbN/deps/build.log`
```

```
using Word2Vec, Gadfly
```

```
┌ Info: Precompiling Word2Vec [c64b6f0f-98cd-51d1-af78-58ae84944834]
└ @ Base loading.jl:1278
┌ Info: Precompiling Gadfly [c91e804a-d5a3-530f-b6f0-dfbca275c004]
└ @ Base loading.jl:1278
```

Opening the text file and making a text vector file in which every word is represented by a number

```
word2vec("text8", "text8-vec.txt", verbose=true)
```

```
Starting training using file text8
Vocab size: 27172
Words in train file: 2901992
Alpha: 0.000002  Progress: 100.23%  Words/thread/sec: 404.06k  Process(`/root/.julia/art
```

This code is to make a file called text8phrase.

```
word2phrase("text8", "text8phrase")
word2vec("text8phrase", "text8phrase-vec.txt", verbose=true)
```

```
tcmalloc: large alloc 2000003072 bytes == 0x15f8000 @  0x7fdaa5492001 0x400b68 0x7fdaa4a
Starting training using file text8

Vocab size (unigrams + bigrams): 2144399
Words in train file: 14492264
Starting training using file text8phrase
Vocab size: 88413
```

```
Words in train file: 13509296
Alpha: 0.000002  Progress: 100.05%  Words/thread/sec: 386.49k  Process(`/root/.julia/art
```

This code is to make a cluster file in which every code will be given a index number.

```
word2clusters("text8", "text8-class.txt", 100)
```

```
Starting training using file text8
Vocab size: 65191
Words in train file: 14231248
Alpha: 0.020007  Progress: 19.99%  Words/thread/sec: 343.35k  Process(`/root/.julia/art
```

Now its time to perfrom some modelling

```
;ls
```

```
sample_data
text8
text8-class.txt
text8phrase
text8phrase-vec.txt
text8-vec.txt
```

Makiing a model

```
model = wordvectors("text8-vec.txt")
```

```
WordVectors 27172 words, 100-element Float64 vectors
```

```
size(model)
```

```
(100, 27172)
```

```
words = vocabulary(model)
```

```
27172-element Array{String,1}:
 "</s>"
 "the"
 "of"
 "and"
 "in"
 "one"
 "a"
 "to"
```

```
 "zero"
 "nine"
 "is"
 "two"
 "as"
 ⋮
 "soundex"
 "shakta"
 "fanu"
 "corces"
 "echinus"
 "adenylate"
 "cimabue"
 "minniti"
 "caravaggisti"
 "gentileschi"
 "prestwich"
 "mousepad"
```

Finding out how many words are similar to love

```
idx = index(model, "love")
```

```
702
```

```
words[idx]
```

```
"love"
```

Finding the point for words similar to son

```
get_vector(model,"son")
```

```
100-element Array{Float64,1}:
 -0.07605775833921728
  0.05554317498138358
  0.041870892208249945
 -0.2049525295138467
  0.002875113688611808
  0.11744093681220638
  0.06157824158304747
  0.04287171452645962
  0.06709470370188979
  0.16065965108580516
  0.16121028339891974
  0.05654323924443838
 -0.10963979121349723
  ⋮
  0.07946076268629776
  0.10093182213578158
 -0.05265498484135035
  0.07608609065063035
```

```
-0.18608908504020197
 0.06465746162210652
 0.15610326582059855
 0.021214931564034027
-0.08143672820434608
-0.06835241196070413
-0.27266514292384536
 0.020277785878832957
```

Finding the similarity between 2 words for example son and daughter

```
similarity(model, "son", "daughter")
```

```
0.9398814174219126
```

Finding the similarity between son and something as random as soccer. The result when compared to the one above will that they are very dissimilar

```
similarity(model, "son", "soccer")
```

```
0.19494015264946568
```

Now im going to find points of the 15 most similar words to soccer and plot them

```
idxs, dists = cosine(model, "soccer", 15)
```

```
([5814, 3846, 3891, 506, 4504, 3732, 2497, 1101, 3870, 2670, 6251, 6553, 3359, 14540, 73
```

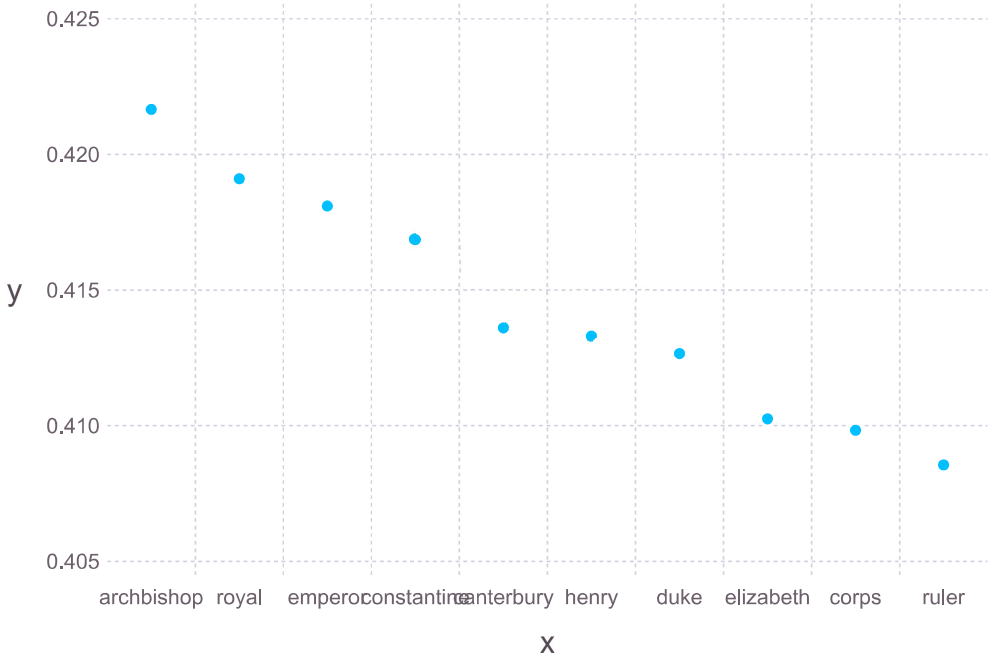
```
plot(x=words[idxs], y=dists)
```

ANALOGY SECTION In this section ill be computing the similarities between words. For example husband - man + wife = woman for example> Ill also plot the result


```
indxs, dists = analogy(model, ["king", "queen"], ["man"], 10)
```

```
([4526, 777, 873, 1820, 4501, 964, 2677, 3033, 2524, 3401], [0.4216602954876769, 0.41916
```

```
plot(x=words[indxs], y=dists)
```



we can also repeat the section above using analogy words which will suggest similar

```
analogy_words(model, ["husband", "wife"], ["man"], 10)
```

```
10-element Array{String,1}:
 "brother"
 "colleague"
 "cousin"
 "qimei"
 "niece"
 "uncle"
```

```
 "sons"
 "daughter"
 "daughters"
 "cicero"
```

```
analogy_words(model, ["england", "italy"], ["london"], 10)
```

```
10-element Array{String,1}:
 "spain"
 "germany"
 "britain"
 "rome"
 "russia"
 "france"
 "greece"
 "romans"
 "rebellion"
 "invaded"
```

Finding out similar phrase usinf the phrase to vec txt

```
model2 = wordvectors("text8phrase-vec.txt")
```

```
WordVectors 88413 words, 100-element Float64 vectors
```

Now im going to use the cosine similar function to find the similar words between words similar to las vegas

```
cosine_similar_words(model2, "las_vegas", 13)
```

```
13-element Array{String,1}:
 "las_vegas"
 "san_diego"
 "san_francisco"
 "atlanta"
 "seattle"
 "los_angeles"
 "dallas"
 "miami"
 "hotel"
 "melbourne"
 "chicago_illinois"
 "st_louis"
 "cleveland"
```

Now im going to try it for a phrase that doesnt exist and it will show that the word cant be found i the song

```
cosine_similar_words(model2, "new_england", 13)
```

```
KeyError: key "new_england" not found

Stacktrace:
 [1] getindex at ./dict.jl:467 [inlined]
 [2] get_vector at /root/.julia/packages/Word2Vec/CY3qd/src/wordvectors.jl:63 [inlined]
 [3] cosine at /root/.julia/packages/Word2Vec/CY3qd/src/wordvectors.jl:74 [inlined]
 [4] cosine_similar_words(::WordVectors{String,Float64,Int64}, ::String, ::Int64) at
/root/.julia/packages/Word2Vec/CY3qd/src/wordvectors.jl:98
 [5] top-level scope at In[29]:1
 [6] include_string(::Function, ::Module, ::String, ::String) at ./loading.jl:1091
```

SEARCH STACK OVERFLOW

CLUSTERING

```
model3 = wordclusters("text8-class.txt")
```

```
WordClusters 65191 words, 100 clusters
```

```
clusters(model3)
```

```
100-element Array{Int64,1}:
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
  ⋮
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
```

We can use the function above to return the cluster ID of a given word for example london, paris, italy

```
get_cluster(model3, "london")
```

```
86
```

```
get_cluster(model3, "paris")
```

```
87
```

```
get_cluster(model3, "italy")
```

```
63
```

We can also find out the cluster ID that are similar to words for example im going to use london which is index 86

```
get_words(model3, 86)
```

```
149-element Array{String,1}:
 "university"
 "york"
 "house"
 "school"
 "home"
 "london"
 "college"
 "press"
 "royal"
 "california"
 "county"
 "park"
 "washington"
 ⋮
 "stonyhurst"
 "northfield"
 "pusey"
 "feliz"
 "jasenovac"
 "cornhuskers"
 "albertson"
 "swarthmore"
 "brixham"
 "benning"
 "mullen"
 "kennett"
```

```
get_words(model3, 35)
```

```
304-element Array{String,1}:
 "see"
 "isbn"
 "references"
 "est"
 "births"
 "http"
 "www"
 "na"
 "jpg"
 "pp"
 "vol"
 "html"
 "iso"
 ⋮
 "cotopaxi"
 "escwa"
 "peloza"
 "scaleminor"
 "pinkerton"
 "biologie"
 "bivouac"
 "wissenschaftslehre"
 "frogmen"
 "guettard"
 "westfall"
 "icehenge"
```

TEXT ANALYSIS & NATURAL LANGUAGE PROCESSING

```
Pkg.add("TextAnalysis")
```

```
  Resolving package versions...
  Installed IniFile ──────────────────── v0.5.0
  Installed InvertedIndices ──────────── v1.0.0
  Installed PooledArrays ─────────────── v0.5.3
  Installed StrTables ────────────────── v1.0.1
  Installed IteratorInterfaceExtensions ─ v1.0.0
  Installed DataDeps ─────────────────── v0.7.4
  Installed TextAnalysis ─────────────── v0.7.1
  Installed DataFrames ───────────────── v0.21.8
  Installed Snowball_jll ─────────────── v2.0.0+0
  Installed Snowball ─────────────────── v0.1.0
  Installed DataValueInterfaces ──────── v1.0.0
  Installed TableTraits ──────────────── v1.0.0
  Installed Tables ───────────────────── v1.2.1
  Installed URIs ─────────────────────── v1.1.0
  Installed HTML_Entities ────────────── v1.0.0
  Installed Languages ────────────────── v0.4.3
  Installed WordTokenizers ───────────── v0.5.6
  Installed HTTP ─────────────────────── v0.9.0
######################################################################## 100.0%
Updating `~/.julia/environments/v1.5/Project.toml`
  [a2db99b7] + TextAnalysis v0.7.1
Updating `~/.julia/environments/v1.5/Manifest.toml`
```

```
  [124859b0] + DataDeps v0.7.4
  [a93c6f00] + DataFrames v0.21.8
  [e2d170a0] + DataValueInterfaces v1.0.0
  [7693890a] + HTML_Entities v1.0.0
  [cd3eb016] + HTTP v0.9.0
  [83e8ac13] + IniFile v0.5.0
  [41ab1584] + InvertedIndices v1.0.0
  [82899510] + IteratorInterfaceExtensions v1.0.0
  [8ef0a80b] + Languages v0.4.3
  [2dfb63ee] + PooledArrays v0.5.3
  [fb8f903a] + Snowball v0.1.0
  [88f46535] + Snowball_jll v2.0.0+0
  [9700d1a9] + StrTables v1.0.1
  [3783bdb8] + TableTraits v1.0.0
  [bd369af6] + Tables v1.2.1
  [a2db99b7] + TextAnalysis v0.7.1
  [5c2747f8] + URIs v1.1.0
  [796a5d58] + WordTokenizers v0.5.6
   Building HTML_Entities → `~/.julia/packages/HTML_Entities/g4t7p/deps/build.log`
```

```
Pkg.add("WordTokenizers")
```

```
  Resolving package versions...
Updating `~/.julia/environments/v1.5/Project.toml`
  [796a5d58] + WordTokenizers v0.5.6
No Changes to `~/.julia/environments/v1.5/Manifest.toml`
```

```
using TextAnalysis, WordTokenizers
```

```
┌ Info: Precompiling TextAnalysis [a2db99b7-8b79-58f8-94bf-bbc811eef33d]
└ @ Base loading.jl:1278
```

Readind the text8 document as a string. This way we can know what the file consists of and ehat language is used to write it.

```
fd = FileDocument("text8")
```

```
A FileDocument
 * Language: Languages.English()
 * Title: text8
 * Author: Unknown Author
 * Timestamp: Unknown Time
 * Snippet:  anarchism originated as a term of abuse first use
```

Tokenization with Txt Analysis The means we are spilt the sentences into words in order to make feature from it if we choose too.

```
tokens(fd)
```

```
14492265-element Array{String,1}:
```

```
 "anarchism"
 "originated"
 "as"
 "a"
 "term"
 "of"
 "abuse"
 "first"
 "used"
 "against"
 "early"
 "working"
 "class"
 ⋮
 "extensively"
 "excavated"
 "in"
 "one"
 "nine"
 "six"
 "three"
 "one"
 "nine"
 "six"
 "five"
 "b"
```

Now im going to to Tokenize a string with WordTokenizer

```
using WordTokenizers
```

```
tokenize("My name is Divine and I am from Nigeria")
```

```
9-element Array{String,1}:
 "My"
 "name"
 "is"
 "Divine"
 "and"
 "I"
 "am"
 "from"
 "Nigeria"
```

Now im going to token a text file i made in order to analyse it.

```
myfiles = FileDocument("docker instructions.txt")
```

```
A FileDocument
 * Language: Languages.English()
 * Title: docker instructions.txt
 * Author: Unknown Author
```

```
 * Timestamp: Unknown Time
 * Snippet: First step is to build a docker build on your loca
```

```
text(myfiles)
```

```
"First step is to build a docker build on your local machine . Then run the docker in a
```

Tokenizing the sentences in the file

```
split_sentences(text(myfiles))
```

```
10-element Array{SubString{String},1}:
 "First step is to build a docker build on your local machine ."
 "Then run the docker in a container and check if its working."
 "After that I pushed the docker file to my GitHub repo."
 "The next was to create a VM instance on google cloud console and specify the machine r
 "After that I opened the terminal on the google console then I cloned my repo from GitH
 "Then I checked to see if the docker file was on my repo."
 "After that Installed the docker certificates on the cloud shell."
 "Then you get the docker form the remote repo, build it and run docker."
 "After doing that cloud shell will give you your ip address which should be running you
 " "
```

This function tell you how many times a world was used

```
ngrams(myfiles)
```

```
Dict{String,Int64} with 72 entries:
  "build"        => 3
  "pushed"       => 1
  "instance"     => 1
  "cloned"       => 1
  "was"          => 2
  "give"         => 1
  "doing"        => 1
  "step"         => 1
  "create"       => 1
  "that"         => 4
  "will"         => 1
  "."            => 2
  "opened"       => 1
  "VM"           => 1
  "google"       => 2
  "running"      => 1
  "certificates" => 1
  "to"           => 5
  "working."     => 1
  "requirements" => 1
  "is"           => 1
```

```
"and"          => 3
"First"        => 1
"address"      => 1
"GitHub"       => 1
⋮              => ⋮
```