InvertedIndex

Fall 2022, CIS 612 Lab 4 Bradley Dowling, 2657649

Setup

This project is implemented in Julia and Python using PostgreSQL 15 as the database. Tested on Ubuntu 22.04 x86_64 with Julia 1.8.2, and Python >= 3.7 with NLTK installed. The code is available at https://github.com/hairshirt/InvertedIndex.jl. It has a command line interface that can be used with `julia -e "InvertedIndex.jl; julia_main()"` and can be compiled to a standalone binary using PackageCompiler.jl.

Database Connection

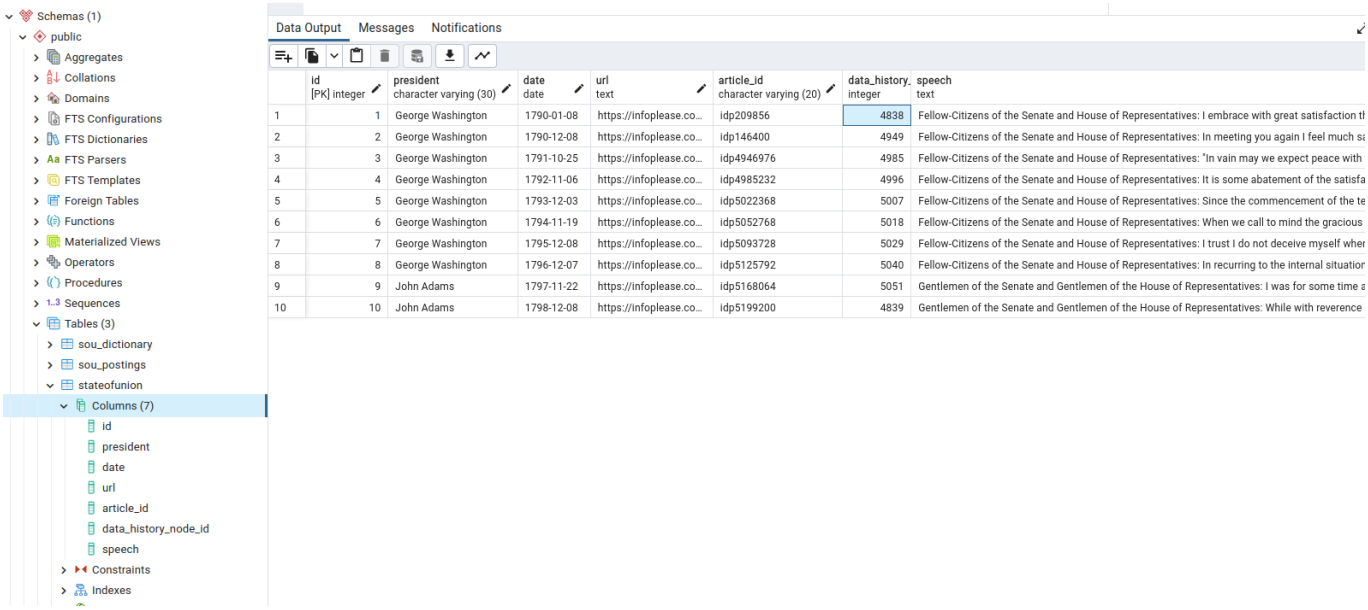
The State of the Union speeches are loaded into the data base from in memory dataframe objects and retrieved using LibPQ as the connection library.

```
const CONN = Ref{LibPQ.Connection}()

connect(user, pass, host, port, dbname) = LibPQ.Connection("dbname=$dbname
user=$user password=$pass port=$port host=$host")

function get_table(conn, table; columns=["*"])
    columns = length(columns) > 1 ? join(columns, ",") : first(columns)
    q = """
        SELECT $(columns) from $(table);
    """
    LibPQ.execute(conn, q) |> DataFrame
end

function load_table(conn, df, table; column_defs=nothing)
    if isnothing(column_defs)
        column_defs = [string(name, " TEXT") for name in names]
    end
    _ = create_table(conn, table, column_defs)
    dropmissing!(df) # just in case
    row_strings = [string(join(collect(row), ','), "\n") for row in
eachrow(df)]
    copyin = LibPQ.CopyIn("COPY $table FROM STDIN (FORMAT CSV);",
row_strings)
    LibPQ.execute(conn, copyin)
end

function create_table(conn, table, columns)
    q = """
        DROP TABLE IF EXISTS $(table);
        CREATE TABLE IF NOT EXISTS $(table)(
```

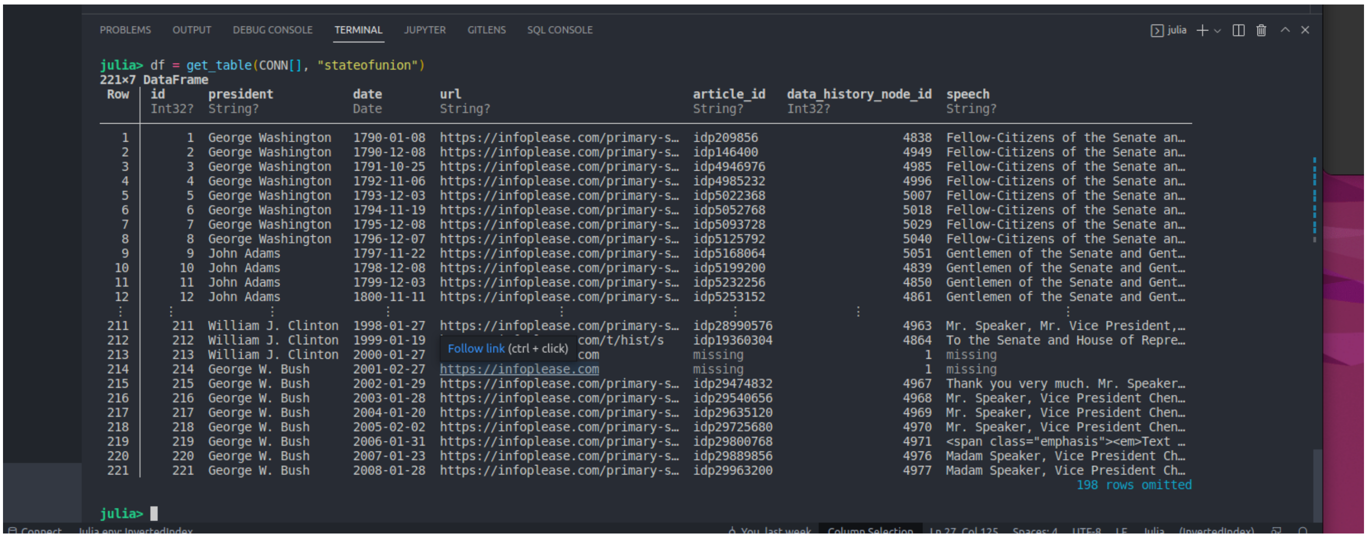
```
            $(join(columns, ",\n"))
        );
    """
    LibPQ.execute(conn, q)
end
```

```
  target_session_attrs = any

julia> const CONN = Ref(ans)
Base.RefValue{LibPQ.Connection}(PostgreSQL connection (CONNECTION_OK) with parameters:
  user = postgres
  password = ********************
  channel_binding = prefer
  dbname = postgres
  host = localhost
  port = 5432
  client_encoding = UTF8
  options = -c DateStyle=ISO,YMD -c IntervalStyle=iso_8601 -c TimeZone=UTC
  application_name = LibPQ.jl
  sslmode = prefer
  sslcompression = 0
  sslsni = 1
  ssl_min_protocol_version = TLSv1.2
  gssencmode = prefer
  krbsrvname = postgres
  target_session_attrs = any)

julia>
```

Here is how they appear in PgAdmin4.



And pulling them back into a dataframe from the database.



```
julia> df = get_table(CONN[], "stateofunion")
221×7 DataFrame
 Row │ id      president           date        url                                article_id   data_history_node_id  speech
     │ Int32?  String?             Date        String?                            String?      Int32?                String?
─────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   1 │      1  George Washington   1790-01-08  https://infoplease.com/primary-s…  idp209856                    4838  Fellow-Citizens of the Senate an…
   2 │      2  George Washington   1790-12-08  https://infoplease.com/primary-s…  idp146400                    4949  Fellow-Citizens of the Senate an…
   3 │      3  George Washington   1791-10-25  https://infoplease.com/primary-s…  idp4946976                   4985  Fellow-Citizens of the Senate an…
   4 │      4  George Washington   1792-11-06  https://infoplease.com/primary-s…  idp4985232                   4996  Fellow-Citizens of the Senate an…
   5 │      5  George Washington   1793-12-03  https://infoplease.com/primary-s…  idp5022368                   5007  Fellow-Citizens of the Senate an…
   6 │      6  George Washington   1794-11-19  https://infoplease.com/primary-s…  idp5052768                   5018  Fellow-Citizens of the Senate an…
   7 │      7  George Washington   1795-12-08  https://infoplease.com/primary-s…  idp5093728                   5029  Fellow-Citizens of the Senate an…
   8 │      8  George Washington   1796-12-07  https://infoplease.com/primary-s…  idp5125792                   5040  Fellow-Citizens of the Senate an…
   9 │      9  John Adams          1797-11-22  https://infoplease.com/primary-s…  idp5168064                   5051  Gentlemen of the Senate and Gent…
  10 │     10  John Adams          1798-12-08  https://infoplease.com/primary-s…  idp5199200                   4839  Gentlemen of the Senate and Gent…
  11 │     11  John Adams          1799-12-03  https://infoplease.com/primary-s…  idp5232256                   4850  Gentlemen of the Senate and Gent…
  12 │     12  John Adams          1800-11-11  https://infoplease.com/primary-s…  idp5253152                   4861  Gentlemen of the Senate and Gent…
  ⋮  │   ⋮             ⋮               ⋮                       ⋮                       ⋮                ⋮                         ⋮
 211 │    211  William J. Clinton  1998-01-27  https://infoplease.com/primary-s…  idp28990576                  4963  Mr. Speaker, Mr. Vice President,…
 212 │    212  William J. Clinton  1999-01-19  Follow link (ctrl + click) com/t/hist/s  idp19360304            4864  To the Senate and House of Repre…
 213 │    213  William J. Clinton  2000-01-27                           com  missing                         1  missing
 214 │    214  George W. Bush      2001-02-27  https://infoplease.com             missing                         1  missing
 215 │    215  George W. Bush      2002-01-29  https://infoplease.com/primary-s…  idp29474832                  4967  Thank you very much. Mr. Speaker…
 216 │    216  George W. Bush      2003-01-28  https://infoplease.com/primary-s…  idp29540656                  4968  Mr. Speaker, Vice President Chen…
 217 │    217  George W. Bush      2004-01-20  https://infoplease.com/primary-s…  idp29635120                  4969  Mr. Speaker, Vice President Chen…
 218 │    218  George W. Bush      2005-02-02  https://infoplease.com/primary-s…  idp29725680                  4970  Mr. Speaker, Vice President Chen…
 219 │    219  George W. Bush      2006-01-31  https://infoplease.com/primary-s…  idp29800768                  4971  <span class="emphasis"><em>Text …
 220 │    220  George W. Bush      2007-01-23  https://infoplease.com/primary-s…  idp29889856                  4976  Madam Speaker, Vice President Ch…
 221 │    221  George W. Bush      2008-01-28  https://infoplease.com/primary-s…  idp29963200                  4977  Madam Speaker, Vice President Ch…
                                                                                                                              198 rows omitted

julia>
```

Build Dictionary and Postings Tables

The dictionary and postings tables are built by sanitizing and stemming all of the input text and collecting all unique terms into an accumulator. The term and document frequency is tracked, and then the tables are loaded into the database. Tf-idf is used as the weight, and several options are available for methods of calculating tf and idf.

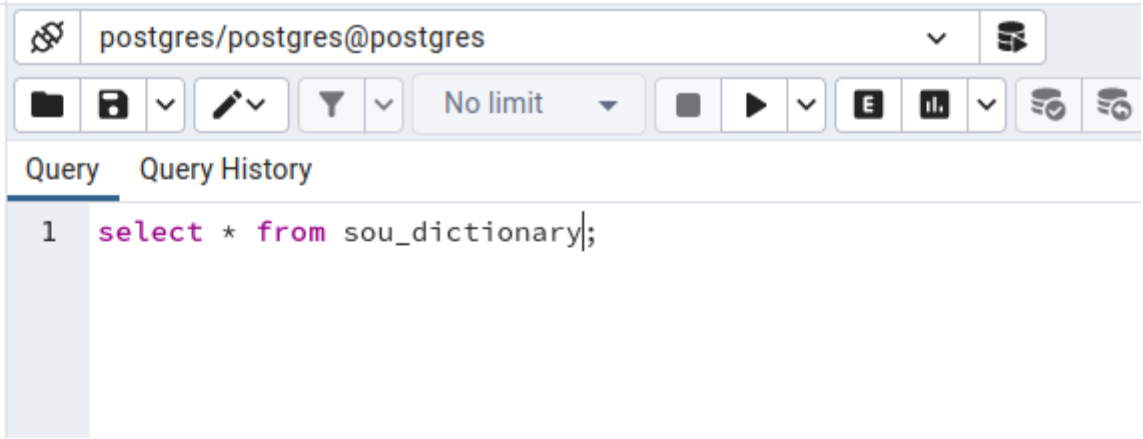
```
const TERM_FREQUENCIES = Ref(Dict{AbstractString,Accumulator}())

function build_inverted_index(df; id_col1=:president, id_col2=:date,
text_col=:speech, tf_method=relative_freq,
idf_method=inv_doc_freq_smooth)::NTuple{2,DataFrame}
    @info "initial df size" size(df)
    dropmissing!(df, [id_col1, id_col2, text_col])
    @info "Dropped missings: " size(df)

    isempty(df) && return df

    doc_ids = string.(df[!, id_col1], "_", df[!, id_col2])
    @info "doc_ids" length(doc_ids)

    documents = replace.(ch -> (isascii(first(ch)) && isletter(first(ch)))
? ch : " ", split.(lowercase.(df[!, text_col]), "")) .|> join
    @info "sanitize documents" length(documents)

    coll_freq = join(documents, ' ') |> sanitize_text |> counter
    @info "Collection Frequency" length(coll_freq)

    terms = collect(keys(coll_freq))
    @info "Unique terms" length(terms)

    dictionary_table = build_dictionary_table(coll_freq, terms, documents;
idf_method=idf_method)
    sort!(unique!(dictionary_table, :term), :term)
    @info "dictionary table" size(dictionary_table)
```

```
    postings_table = build_postings_table(doc_ids, documents;
tf_method=tf_method)
    sort!(unique!(postings_table, [:doc_id, :term]), [:doc_id, :term])
    dd = Dict(row.term => row.idf for row in eachrow(dictionary_table))
    postings_table[!, :tfidf] = [dd[row.term] * row.tf for row in
eachrow(postings_table)]
    @info "postings table" size(postings_table)

    dictionary_table, postings_table
end

function build_postings_table(doc_ids, documents;
tf_method=relative_freq)::DataFrame
    postings = Dict(:term => String[], :doc_id => String[], :termfreq =>
Int64[], :tf => Float64[])
    for (doc_id, document) in zip(doc_ids, documents)
        if !haskey(TERM_FREQUENCIES[], doc_id)
            TERM_FREQUENCIES[][doc_id] = sanitize_text(document) |> counter
        end
        term_freq = TERM_FREQUENCIES[][doc_id]
        for (term, freq) in term_freq
            push!(postings[:term], term)
            push!(postings[:doc_id], doc_id)
            push!(postings[:termfreq], freq)
            push!(postings[:tf], tf(term, term_freq; fn=tf_method))
        end
    end
    return DataFrame(postings)
end


function build_dictionary_table(coll_freq, terms, documents;
idf_method=inv_doc_freq_smooth)::DataFrame
    doc_freq = document_frequency(terms, Set.(split.(documents)))
    DataFrame(Dict(
        :term => terms,
        :docfreq => [doc_freq[term] for term in terms],
        :idf => [idf(term, doc_freq, length(documents), fn=idf_method) for
term in terms],
        :collectionfreq => [coll_freq[term] for term in terms],
    ))
end

function document_frequency(terms, documents)::Dict{String,Int}
    doc_freq = Dict{String,Int}()
    for term in terms
        if !haskey(doc_freq, term)
            push!(doc_freq, term => 0)
        end
        for doc in documents
            if term ∈ doc
                doc_freq[term] += 1
            end
        end
```

```
    end
    return doc_freq
end


tf(term, term_freq; fn=relative_freq)::Float64 = fn(term, term_freq)


augmented(term, term_freq)::Float64 = 0.5 + 0.5 * term_freq[term] /
maximum(values(term_freq))
log_scaled(term, term_freq)::Float64 = log10(1.0 + term_freq[term])
boolean_freq(term, term_freq)::Float64 = iszero(term_freq[term]) ? 0.0 :
1.0
raw_count(term, term_freq)::Float64 = term_freq[term]
relative_freq(term, term_freq)::Float64 = term_freq[term] /
(sum(values(term_freq)) - term_freq[term])

const TF_METHODS = Dict{String,Function}(
    "augmented" => augmented,
    "log_scaled" => log_scaled,
    "boolean_freq" => boolean_freq,
    "raw_count" => raw_count,
    "relative_freq" => relative_freq,
)


idf(term, doc_freq, ndocs; fn=inv_doc_freq_smooth)::Float64 = fn(term,
doc_freq, ndocs)

unary(term, doc_freq, ndocs)::Float64 = iszero(doc_freq[term]) ? 0.0 : 1.0
inv_doc_freq(term, doc_freq, ndocs)::Float64 = log10(ndocs /
doc_freq[term])
inv_doc_freq_smooth(term, doc_freq, ndocs)::Float64 = log10(ndocs / (1.0 +
doc_freq[term])) + 1.0
inv_doc_freq_max(term, doc_freq, ndocs)::Float64 =
log10(maximum(values(doc_freq)) / (1.0 + doc_freq[term]))
probabilistic_inv_doc_freq(term, doc_freq, ndocs)::Float64 = log10((ndocs -
doc_freq[term]) / doc_freq[term])

const IDF_METHODS = Dict{String,Function}(
    "unary" => unary,
    "inv_doc_freq" => inv_doc_freq,
    "inv_doc_freq_smooth" => inv_doc_freq_smooth,
    "inv_doc_freq_max" => inv_doc_freq_max,
    "probabilistic_inv_doc_freq" => probabilistic_inv_doc_freq,
)

function __init__()
    py"""
    from nltk.corpus import stopwords
    from nltk.stem import PorterStemmer
    from nltk.tokenize import word_tokenize
```

```
    def sanitize_text(text: str) -> list[str]:
        '''
        Removes english stop words and suffixes from a string using nltk.
        May require nltk.download('stopwords') and nltk.download('punkt').

        Parameters:
        text (str): A string

        Returns:
        list[str]: A list of word stems with stop words removed
        '''
        stemmer = PorterStemmer()
        stop_words = set(stopwords.words('english'))
        words = list(filter(lambda word: word not in stop_words,
word_tokenize(text)))
        return list(map(lambda word: stemmer.stem(word), words))
    """
end

sanitize_text(text) = py"sanitize_text"(text)

sanitize_string(s) = replace(ch -> (isascii(first(ch)) &&
isletter(first(ch))) ? ch : " ", split(lowercase(s), "")) |> join |> strip
```

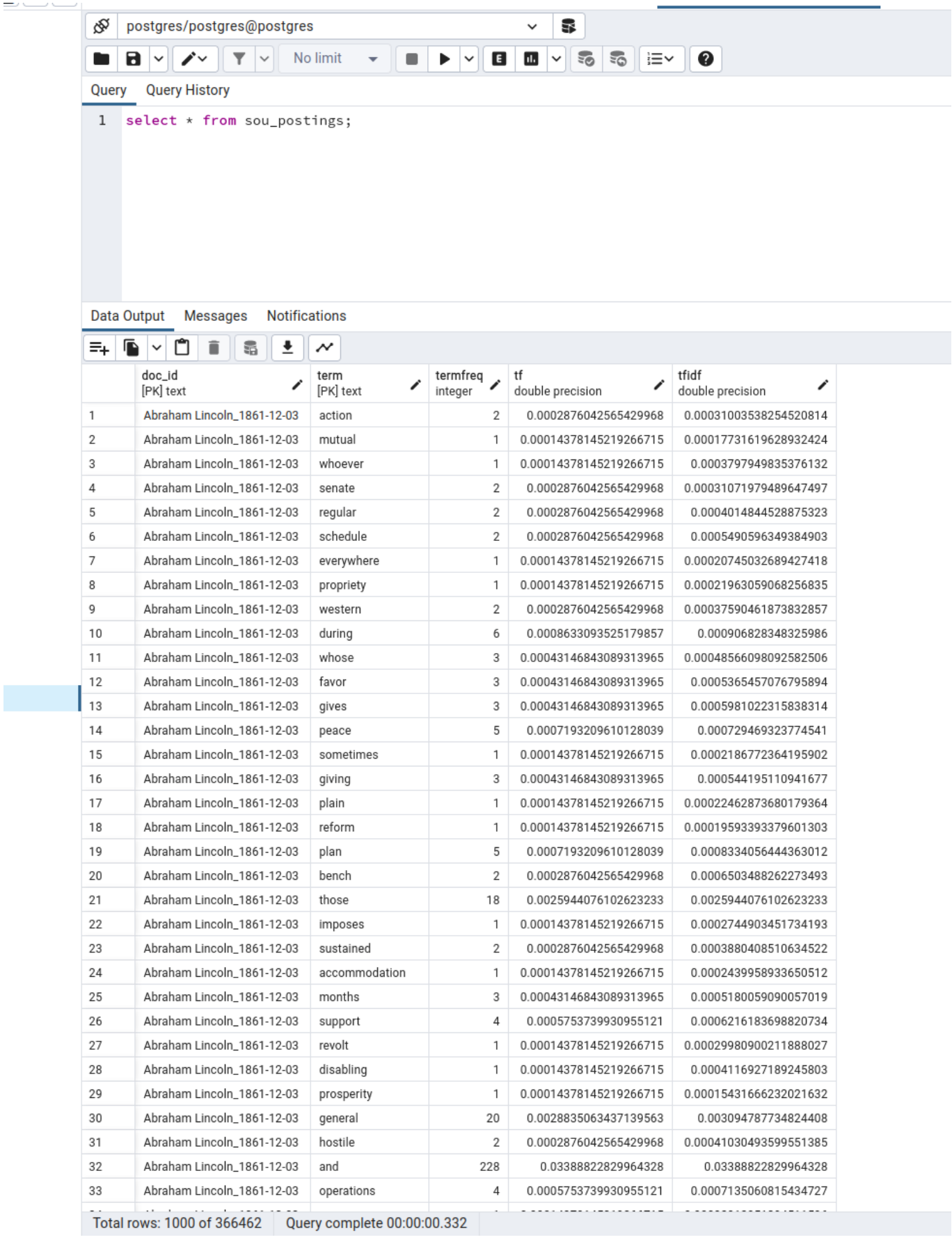
```
julia> dictionary_table, postings_table = build_inverted_index(df)
[ Info: initial df size
  size(df) = (221, 7)
[ Info: Dropped missings:
  size(df) = (219, 7)
[ Info: doc_ids
  length(doc_ids) = 219
[ Info: sanitize documents
  length(documents) = 219
[ Info: Collection Frequency
  length(coll_freq) = 12493
[ Info: Unique terms
  length(terms) = 12493
[ Info: dictionary table
  size(dictionary_table) = (12493, 4)
[ Info: postings table
  size(postings_table) = (263678, 5)
```



Data Output Messages Notifications

	collectionfreq integer	docfreq integer	idf double precision	term [PK] text
1	26558	218	1	a
2	1	1	3.0394141191761372	aaa
3	2	1	3.0394141191761372	aana
4	1	1	3.0394141191761372	aaron
5	78	58	1.5695921031979743	abandon
6	82	60	1.5551142798293514	abandoned
7	13	13	2.1943160791618803	abandoning
8	29	25	1.9254707668693003	abandonment
9	2	2	2.863322860120456	abandons
10	4	4	2.6414741105040997	abate
11	4	4	2.6414741105040997	abated
12	5	4	2.6414741105040997	abatement
13	2	2	2.863322860120456	abating
14	1	1	3.0394141191761372	abbas
15	1	1	3.0394141191761372	abbreviation
16	1	1	3.0394141191761372	abdicate
17	2	1	3.0394141191761372	abdicated
18	2	2	2.863322860120456	abdicating
19	6	6	2.4953460748258616	abdication
20	1	1	3.0394141191761372	abducted
21	1	1	3.0394141191761372	abduction
22	1	1	3.0394141191761372	aberdeen
23	2	2	2.863322860120456	abet
24	1	1	3.0394141191761372	abetted
25	2	2	2.863322860120456	abettors
26	9	8	2.3862016054007933	abeyance
27	1	1	3.0394141191761372	abhor
28	3	3	2.738384123512156	abhorrence
29	2	2	2.863322860120456	abhorrent
30	3	3	2.738384123512156	abhors
31	16	15	2.1363241321841935	abide
32	1	1	3.0394141191761372	abides
33	1	1	3.0394141191761372	abideth

Total rows: 1000 of 22548 Query complete 00:00:00.191



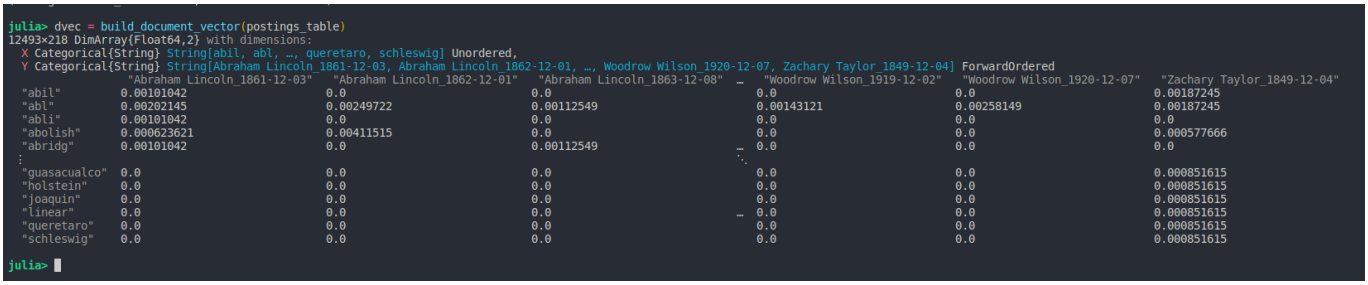
	doc_id [PK] text	term [PK] text	termfreq integer	tf double precision	tfidf double precision
1	Abraham Lincoln_1861-12-03	action	2	0.0002876042565429968	0.00031003538254520814
2	Abraham Lincoln_1861-12-03	mutual	1	0.0001437814521926671 5	0.00017731619628932424
3	Abraham Lincoln_1861-12-03	whoever	1	0.0001437814521926671 5	0.0003797949835376132
4	Abraham Lincoln_1861-12-03	senate	2	0.0002876042565429968	0.00031071979489647497
5	Abraham Lincoln_1861-12-03	regular	2	0.0002876042565429968	0.0004014844528875323
6	Abraham Lincoln_1861-12-03	schedule	2	0.0002876042565429968	0.0005490596349384903
7	Abraham Lincoln_1861-12-03	everywhere	1	0.0001437814521926671 5	0.00020745032689427418
8	Abraham Lincoln_1861-12-03	propriety	1	0.0001437814521926671 5	0.00021963059068256835
9	Abraham Lincoln_1861-12-03	western	2	0.0002876042565429968	0.00037590461873832857
10	Abraham Lincoln_1861-12-03	during	6	0.0008633093525179857	0.000906828348325986
11	Abraham Lincoln_1861-12-03	whose	3	0.0004314684308931396 5	0.00048566098092582506
12	Abraham Lincoln_1861-12-03	favor	3	0.0004314684308931396 5	0.0005365457076795894
13	Abraham Lincoln_1861-12-03	gives	3	0.0004314684308931396 5	0.0005981022315838314
14	Abraham Lincoln_1861-12-03	peace	5	0.0007193209610128039	0.000729469323774541
15	Abraham Lincoln_1861-12-03	sometimes	1	0.0001437814521926671 5	0.00021867723641959 02
16	Abraham Lincoln_1861-12-03	giving	3	0.0004314684308931396 5	0.000544195110941677
17	Abraham Lincoln_1861-12-03	plain	1	0.0001437814521926671 5	0.00022462873680179364
18	Abraham Lincoln_1861-12-03	reform	1	0.0001437814521926671 5	0.00019593933793601303
19	Abraham Lincoln_1861-12-03	plan	5	0.0007193209610128039	0.0008334056444363012
20	Abraham Lincoln_1861-12-03	bench	2	0.0002876042565429968	0.0006503488262273493
21	Abraham Lincoln_1861-12-03	those	18	0.0025944076102623233	0.0025944076102623233
22	Abraham Lincoln_1861-12-03	imposes	1	0.0001437814521926671 5	0.0002744903451734193
23	Abraham Lincoln_1861-12-03	sustained	2	0.0002876042565429968	0.0003880408510634522
24	Abraham Lincoln_1861-12-03	accommodation	1	0.0001437814521926671 5	0.0002439958933650512
25	Abraham Lincoln_1861-12-03	months	3	0.0004314684308931396 5	0.0005180059090057019
26	Abraham Lincoln_1861-12-03	support	4	0.0005753739930955121	0.0006216183698820734
27	Abraham Lincoln_1861-12-03	revolt	1	0.0001437814521926671 5	0.00029980900211888027
28	Abraham Lincoln_1861-12-03	disabling	1	0.0001437814521926671 5	0.0004116927189245803
29	Abraham Lincoln_1861-12-03	prosperity	1	0.0001437814521926671 5	0.00015431666232021632
30	Abraham Lincoln_1861-12-03	general	20	0.0028835063437139563	0.003094787734824408
31	Abraham Lincoln_1861-12-03	hostile	2	0.0002876042565429968	0.00041030493599551385
32	Abraham Lincoln_1861-12-03	and	228	0.03388822829964328	0.03388822829964328
33	Abraham Lincoln_1861-12-03	operations	4	0.0005753739930955121	0.0007135060815434727

Total rows: 1000 of 366462 Query complete 00:00:00.332

Build Document Vector

Building a matrix of weights for terms and the documents in which they appear.

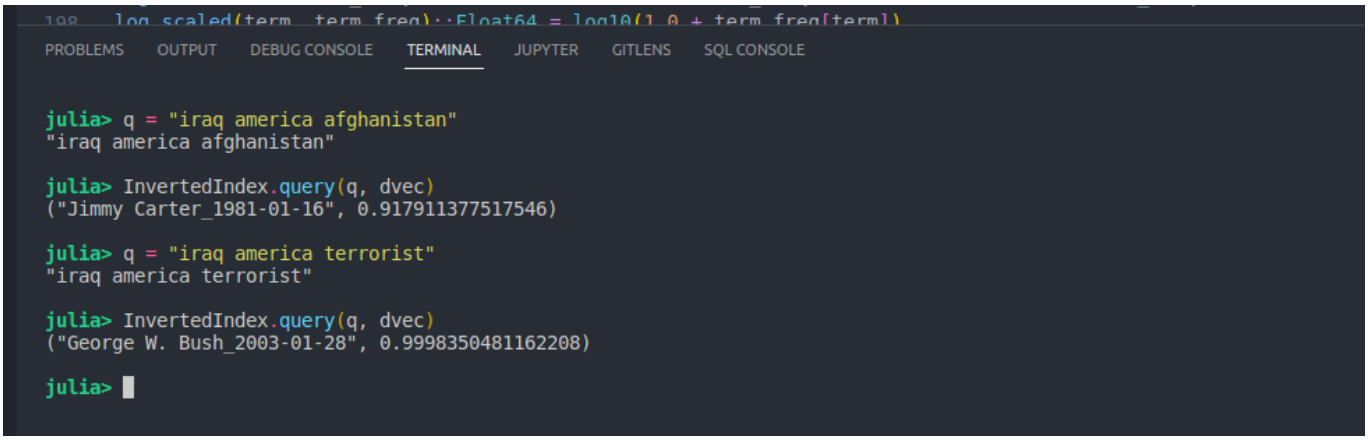
```
function build_document_vector(postings)
    dvec = zeros(X(unique(postings.term)), Y(unique(postings.doc_id)))
    for row in eachrow(postings)
        dvec[X(At(row.term)), Y(At(row.doc_id))] = row.tfidf
    end
    return dvec
end
```



Query Results

Query results are computed by cosine similarity on common terms between the document and the query

```
cosine_similarity(A, B) = (A · B) / (norm(A) * norm(B))

function query(keywords, dvec)
    isempty(keywords) && return nothing

    keywords = sanitize_string(keywords) |> sanitize_text
    terms, docs = dvec.dims
    keywords = filter(∈(terms), keywords)

    isempty(keywords) && return nothing

    kws = ones(Float64, length(keywords))
    dvec = dvec[X(At(keywords)), Y()]

    sim = 0.0
    idx = nothing
    for i in 1:length(docs)
        s = cosine_similarity(kws, dvec[:, i])
        if s > sim
            sim = s
            idx = i
        end
    end
    return docs[idx], sim
end
```



Full procedure

```
# btd @ Ubuntu-Desktop in ~/.julia/dev/InvertedIndex on git:main x [19:59:02]
$ julia --project=. -e "using InvertedIndex; InvertedIndex.julia_main()" --trace-compile=precompile.jl
┌ Info: CLI args:
│   args =
│    Dict{Symbol, Any} with 17 entries:
│      :search_string    => "freedom of speech"
│      :user             => "postgres"
│      :dictionary       => "sou_dictionary"
│      :idf              => "inv_doc_freq_smooth"
│      :columns          => "*"
│      :text_col         => :speech
│      :host             => "localhost"
│      :postings         => "sou_postings"
│      :upload_to_db     => false
│      :idcol2           => :date
│      :load_idx_from_db => false
│      :db               => "postgres"
│      :tf               => "relative_freq"
│      :table            => "stateofunion"
│      :port             => 5432
│      :pass             => "postgres"
└      :idcol1           => :president
[ Info: Connecting to database
┌ Info: PostgreSQL connection (CONNECTION_OK) with parameters:
│   user = postgres
│   password = ********************
│   channel_binding = prefer
│   dbname = postgres
│   host = localhost
│   port = 5432
│   client_encoding = UTF8
│   options = -c DateStyle=ISO,YMD -c IntervalStyle=iso_8601 -c TimeZone=UTC
│   application_name = LibPQ.jl
│   sslmode = prefer
│   sslcompression = 0
│   sslsni = 1
│   ssl_min_protocol_version = TLSv1.2
│   gssencmode = prefer
│   krbsrvname = postgres
└   target_session_attrs = any
┌ Info: Retrieved SOU table from database
└   size(df) = (221, 7)
[ Info: Building inverted index
┌ Info: initial df size
└   size(df) = (221, 7)
┌ Info: Dropped missings:
└   size(df) = (219, 7)
┌ Info: doc_ids
└   length(doc_ids) = 219
┌ Info: sanitize documents
└   length(documents) = 219
┌ Info: Collection Frequency
└   length(coll_freq) = 12493
┌ Info: Unique terms
└   length(terms) = 12493
┌ Info: dictionary table
└   size(dictionary_table) = (12493, 4)
┌ Info: postings table
└   size(postings_table) = (263678, 5)
[ Info: Building document vector
┌ Info: Search string
└   args[:search_string] = "freedom of speech"
┌ Info: Search result
└   query(args[:search_string], dvec) = ("Grover Cleveland_1895-12-02", 0.995279222463527)
[ Info: Closing database connection
[ Info: PostgreSQL connection (closed)
[ Info: Success

# btd @ Ubuntu-Desktop in ~/.julia/dev/InvertedIndex on git:main x [20:01:22]
$
```