**In the term project report, each team needs to clearly indicate the contributions of each team member.** You should also include detailed descriptions about (1) which methods your team has tried, (2) what's the results of each of those methods, (3) what do you learn from trying a method, and (4) the codes of your best method.

1. Which Methods Your Team Has Tried

We had four methods that we tried for our final project. First, we generated the code for random results with Java. After the random results, we used serious computations in Matlab, which are labeled “PROGRESS 1”. After this we tried to apply a neural network to our code. However, we were unable to make it work. Finally, we wrote a better serious computational code in Matlab. In this method we used a part of the theory from the article about deep learning. That article was called “Deep learning for answer sentence selection.” The second serious computational code is labeled “PROGESS 2”. We have detail descriptions of the classes made for both PROGRESS 1 and PROGRESS 2 below.

PROGRESS 1.

main.m

This method reads the questions from the input file “training\_set.tsv”. Next it reads the noise.txt file, which contains the prepositions that are irrelevant for the feature extraction process. The words that are in the noise.txt file are then removed from the questions so that only the relevant words are left for feature extraction. We also remove punctuations. We repeat this process with the answers as well. This leaves us with only the relevant words for the questions and answers.

string2hash.m

This function converts strings into hash values in order to have each question and answer represented as a vector. The norm of the vector for the question is noted and the corresponding answer is also noted from the given label.

training.m

This code reads the questions and answers from the input file “validation\_set.tsv”. It performs the same tasks performed by “main.m”. However, it does not note the corresponding answer as it does not have any given label. Then there is the most important part that this code accomplishes.

Let’s say the question from the validation set is represented by Y with the corresponding answer choices represented by vectors Y\_a, Y\_b, Y\_c, and Y\_d. We search for the vector closest to Y in the training set. Now let’s say that vector is represented by X. We know the corresponding answer of X from the training set. This is represented by X\_x. Now we search through Y\_a, Y\_b, Y\_c, and Y\_d for which is closest to X\_x. For example, if Y\_b is the closest vector to X\_x, then we can conclude that the answer of Y is Y\_b.

We then write the question id and corresponding answer option to the file “submission.csv”.

PROGRESS 2.

main\_new.m

This does the same thing as main.m in PROGRESS 1 except that we now create a training matrix. We build the training matrix for the correct answers. If an answer is the correct answer it will be represented by a 1 in the training matrix. If an answer isn’t the correct answer it will represented by a 0 in the training matrix. For example, if C was the correct answer for a given question, then the matrix would be [0 0 1 0]. We will call this matrix, M. We also have a matrix of all the answers. This is a matrix of the vectors of the answer strings. We will call this matrix, A. This gives us a learning matrix, M\*A, which will be useful later.

string2hash.m

This does the same thing as it did in PROGRESS 1.

training\_new.m

This does the same thing as the training.m code from PROGRESS 1 except that we now find the semantic similarity between the training questions and test questions for each question. To do this, we will choose the most similar question and its corresponding answer options from the test set. Then we need to find the semantic similarity between the correct answer from the training set and the answer option of the corresponding question from the test set. This is where we use the learning matrix, which stored the correct answers during the training phase. We assume that the most similar answer is assumed to be the correct one. Finally, we create the csv file for submission.

1. What's The Results of Each of Those Methods

Our results from random guessing were 0.25500.

Our results from the first serious computation, PROGRESS 1, were 0.22500.

Unfortunately we weren’t able to get a result from the code with the neural network because we couldn’t figure out how to make the code work.

Finally, our results from the second serious computation, PROGRESS 2, were 0.24000.

1. What Do You Learn From Trying A Method

We learned that an A.I. can still have fluctuating results without changing code. We made multiple submissions with PROGRESS 1 yet received different scores. One score was 0.22500 and the other score was 0.22375. The scores kept going back and forth between each other with each submission. It shows that A.I. is very complex because it can change an answer along the lines with each run. This outcome is possibly due to 1 or 2 questions having 2 similar answers that the A.I. cannot choose between. Thus, it picks one and hopes it is right, just like any 8th grader would do.

We also learned the progress that can be made by applying different techniques to the same problem. Every time we applied a new technique to our code we were able to increase our score. By adding the matrices on top of the vectors with PROGRESS 2 we were able to increase our score by 1.5 points.

If we were able to add a neural network to our code and make it work I have no doubt we would be able to increase our score even more. There was also the idea of the A.I. choosing a random answer between 2 answers with the highest scores for a question if they were close. Unfortunately, we were never able to implement that idea. However, we don’t know if it would have worked. This idea came from the thought that if an actual student doesn’t know the answer between two choices they would guess between the 2.

1. The Codes of Your Best Method

We have provided the code for PROGRESS 2 below since it yielded the best results.

*PROGRESS 2.*

**string2hash.m**

function hash=string2hash(str,type)

% This function generates a hash value from a text string

%

% hash=string2hash(str,type);

%

% inputs,

%   str : The text string, or array with text strings.

% outputs,

%   hash : The hash value, integer value between 0 and 2^32-1

%   type : Type of has 'djb2' (default) or 'sdbm'

%

% From c-code on : http://www.cse.yorku.ca/~oz/hash.html

%

% djb2

%  this algorithm was first reported by dan bernstein many years ago

%  in comp.lang.c

%

% sdbm

%  this algorithm was created for sdbm (a public-domain reimplementation of

%  ndbm) database library. it was found to do well in scrambling bits,

%  causing better distribution of the keys and fewer splits. it also happens

%  to be a good general hashing function with good distribution.

%

% example,

%

%  hash=string2hash('hello world');

%  disp(hash);

%

% Function is written by D.Kroon University of Twente (June 2010)

% From string to double array

str=double(str);

if(nargin<2), type='djb2'; end

switch(type)

   case 'djb2'

       hash = 5381\*ones(size(str,1),1);

       for i=1:size(str,2),

           hash = mod(hash \* 33 + str(:,i), 2^32-1);

       end

   case 'sdbm'

       hash = zeros(size(str,1),1);

       for i=1:size(str,2),

           hash = mod(hash \* 65599 + str(:,i), 2^32-1);

       end

   otherwise

       error('string\_hash:inputs','unknown type');

end

**main\_new.m**

A = tdfread('training\_set.tsv','\t');

B=cellstr(A.question);

BA\_1 = cellstr(A.answerA);

BA\_2 = cellstr(A.answerB);

BA\_3 = cellstr(A.answerC);

BA\_4 = cellstr(A.answerD);

for i=1:length(B)

   train.question(i)=string2hash(B{i});%/length(B{i});

   train.answerA(i)=string2hash(BA\_1{i});%/length(BA\_1{i});

   train.answerB(i)=string2hash(BA\_2{i});%/length(BA\_2{i});

   train.answerC(i)=string2hash(BA\_3{i});%/length(BA\_3{i});

   train.answerD(i)=string2hash(BA\_4{i});%/length(BA\_4{i});

end

CA = cellstr(A.correctAnswer);

for i=1:length(CA)

   if CA{i}=='A'

       train.CA\_mat(i,:) = [1 0 0 0];

   elseif CA{i}=='B'

       train.CA\_mat(i,:) = [0 1 0 0];

   elseif CA{i}=='C'

       train.CA\_mat(i,:) = [0 0 1 0];

   elseif CA{i}=='D'

       train.CA\_mat(i,:) = [0 0 0 1];

   end

end

answers = [train.answerA;train.answerB;train.answerC;train.answerD];

for i=1:length(train.question)

   train.learn\_mat(i) = train.CA\_mat(i,:)\*answers(:,i);

end

save train train

**training\_new.m**

load('train.mat')

A = tdfread('validation\_set.tsv','\t');

B=cellstr(A.question);

BA\_1 = cellstr(A.answerA);

BA\_2 = cellstr(A.answerB);

BA\_3 = cellstr(A.answerC);

BA\_4 = cellstr(A.answerD);

for i=1:length(B)

   test.question(i)=string2hash(B{i});%/length(B{i});

   test.answerA(i)=string2hash(BA\_1{i});%/length(BA\_1{i});

   test.answerB(i)=string2hash(BA\_2{i});%/length(BA\_2{i});

   test.answerC(i)=string2hash(BA\_3{i});%/length(BA\_3{i});

   test.answerD(i)=string2hash(BA\_4{i});%/length(BA\_4{i});

end

test.answers = [test.answerA;test.answerB;test.answerC;test.answerD]';

for i=1:length(test.question)

   temp\_q = test.question(i)\*ones(1,length(train.question));

   temp\_diff = abs(temp\_q-train.question);

   [diff index] = min(temp\_diff);

   temp\_ans = train.learn\_mat(index);

   temp\_ans\_diff = abs(temp\_ans\*ones(1,4) - test.answers(1,:));

   [dif idx] = min(temp\_ans\_diff);

   if idx==1

       test.CA\_mat(i)='A';

   elseif idx==2

       test.CA\_mat(i)='B';

   elseif idx==3

       test.CA\_mat(i)='C';

   elseif idx==4

       test.CA\_mat(i)='D';

   end

end

id = A.id;

for i=1:length(A.id)

   id\_str = num2str(id(i));

   submission{i,1}= id\_str;

   submission{i,2}= test.CA\_mat(i);

end

submission1=[{'id','correctAnswer'};submission];

temp=cell2dataset(submission1);

% temp.Properties.VarNames{1} = 'id';

% temp.Properties.VarNames{2} = 'correctAnswer';

export(temp,'file','submission\_new.csv','delimiter',',');

First serious computation is PROGRESS1 and code

*PROGRESS1.*

**CODE FOR main.m**

clear;clc;

A = tdfread('training\_set.tsv','\t');

B=cellstr(A.question);

a=0;

for i=1:size(A.question,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

%             clc;

           k=k+1;

%             a

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           A\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           A\_vec(i,j)=0;

       else

           A\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save A\_vec A\_vec

clear

%%%%% Vectors for questions retrieved.

A = tdfread('training\_set.tsv','\t');

B=cellstr(A.answerA);

a=0;

for i=1:size(A.answerA,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_A\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_A\_vec(i,j)=0;

       else

           Ans\_A\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_A\_vec Ans\_A\_vec

clear

%%%%%%%%%%%%%%%%%%% ans A done

A = tdfread('training\_set.tsv','\t');

B=cellstr(A.answerB);

a=0;

for i=1:size(A.answerB,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_B\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_B\_vec(i,j)=0;

       else

           Ans\_B\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_B\_vec Ans\_B\_vec

clear

%%%%%%%%%%%%%%%%%%%%%%% Ans B done

A = tdfread('training\_set.tsv','\t');

B=cellstr(A.answerC);

a=0;

for i=1:size(A.answerC,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_C\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_C\_vec(i,j)=0;

       else

           Ans\_C\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_C\_vec Ans\_C\_vec

clear

%%%%%%%%%%%%%%%%% Ans C done

A = tdfread('training\_set.tsv','\t');

B=cellstr(A.answerD);

a=0;

for i=1:size(A.answerD,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_D\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_D\_vec(i,j)=0;

       else

           Ans\_D\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_D\_vec Ans\_D\_vec

clear

load('A\_vec.mat')

load('Ans\_A\_vec.mat')

load('Ans\_B\_vec.mat')

load('Ans\_C\_vec.mat')

load('Ans\_D\_vec.mat')

A = tdfread('training\_set.tsv','\t');

for i=1:2500

   ques\_norm(i) = norm(A\_vec(i,:));

   ansA\_norm(i) = norm(Ans\_A\_vec(i,:));

   ansB\_norm(i) = norm(Ans\_B\_vec(i,:));

   ansC\_norm(i) = norm(Ans\_C\_vec(i,:));

   ansD\_norm(i) = norm(Ans\_D\_vec(i,:));

end

for i=1:2500

   if A.correctAnswer(i)=='A'

       label(i)=ansA\_norm(i);

   elseif A.correctAnswer(i)=='B'

       label(i)=ansB\_norm(i);

   elseif A.correctAnswer(i)=='C'

       label(i)=ansC\_norm(i);

   elseif A.correctAnswer(i)=='D'

       label(i)=ansD\_norm(i);

   end

end

save ques\_norm ques\_norm

save label label

%%%%%%%%%%%%%%%%%% Ans D done

**CODE FOR training.m**

clear;clc;

load('label.mat')

load('ques\_norm.mat')

A = tdfread('validation\_set.tsv','\t');

B=cellstr(A.question);

a=0;

for i=1:size(A.question,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

%             clc;

           k=k+1;

%             a

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           A\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           A\_vec(i,j)=0;

       else

           A\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save A\_vec A\_vec

clear

%%%%% Vectors for questions retrieved.

A = tdfread('validation\_set.tsv','\t');

B=cellstr(A.answerA);

a=0;

for i=1:size(A.answerA,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_A\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_A\_vec(i,j)=0;

       else

           Ans\_A\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_A\_vec Ans\_A\_vec

clear

%%%%%%%%%%%%%%%%%%% ans A done

A = tdfread('validation\_set.tsv','\t');

B=cellstr(A.answerB);

a=0;

for i=1:size(A.answerB,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_B\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_B\_vec(i,j)=0;

       else

           Ans\_B\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_B\_vec Ans\_B\_vec

clear

%%%%%%%%%%%%%%%%%%%%%%% Ans B done

A = tdfread('validation\_set.tsv','\t');

B=cellstr(A.answerC);

a=0;

for i=1:size(A.answerC,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_C\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_C\_vec(i,j)=0;

       else

           Ans\_C\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_C\_vec Ans\_C\_vec

clear

%%%%%%%%%%%%%%%%% Ans C done

A = tdfread('validation\_set.tsv','\t');

B=cellstr(A.answerD);

a=0;

for i=1:size(A.answerD,1)

   k=1;

   for j=1:size(B{i},2)

       if B{i}(j) == '.' || B{i}(j) == '?' || B{i}(j) == ',' || B{i}(j) == '\_'

           a=a+1;

       else

           A\_pure(i,k)=B{i}(j);

           k=k+1;

       end

   end

   clc;

   i

end

A\_cell = cellstr(A\_pure);

for i=1:size(A\_cell,1)

    C{i} = strsplit(A\_cell{i});

end

A\_noise = tdfread('noise.txt');

A\_noise = cellstr(A\_noise.aboard);

for i=1:length(C)

   for j=1:size(C{i},2)

       if length(C{i}{j})==0

           Ans\_D\_vec(i,j)=0;

       elseif sum(strcmp(C{i}{j},A\_noise))>0

           Ans\_D\_vec(i,j)=0;

       else

           Ans\_D\_vec(i,j)=string2hash(C{i}{j},'sdbm');

       end

   end

  clc; i

end

save Ans\_D\_vec Ans\_D\_vec

clear

load('A\_vec.mat')

load('Ans\_A\_vec.mat')

load('Ans\_B\_vec.mat')

load('Ans\_C\_vec.mat')

load('Ans\_D\_vec.mat')

load('ques\_norm');

load('label');

for i=1:size(A\_vec,1)

   val\_ques\_norm(i) = norm(A\_vec(i,:));

   ansA\_norm(i) = norm(Ans\_A\_vec(i,:));

   ansB\_norm(i) = norm(Ans\_B\_vec(i,:));

   ansC\_norm(i) = norm(Ans\_C\_vec(i,:));

   ansD\_norm(i) = norm(Ans\_D\_vec(i,:));

end

for i=1:length(val\_ques\_norm)

   mat = val\_ques\_norm(i)\*ones(1,length(ques\_norm));

   diff = abs(ques\_norm-mat);

   [M,IDX] = min(diff);

   prob\_ans = label(IDX)\*ones(1,4);

   ans\_set = [ansA\_norm(i) ansB\_norm(i) ansC\_norm(i) ansD\_norm(i)];

   [ANS,ANS\_IDX] = min(abs(ans\_set-prob\_ans));

   if ANS\_IDX==1

       c\_ans(i)='A';

   elseif ANS\_IDX==2

       c\_ans(i)='B';

   elseif ANS\_IDX==3

       c\_ans(i)='C';

   elseif ANS\_IDX==4

       c\_ans(i)='D';

   end

end

A = tdfread('validation\_set.tsv','\t');

id = A.id;

for i=1:length(A.id)

   id\_str = num2str(id(i));

   submission{i,1}= id\_str;

   submission{i,2}= c\_ans(i);

end

submission1=[{'id','correctAnswer'};submission];

temp=cell2dataset(submission1);

% temp.Properties.VarNames{1} = 'id';

% temp.Properties.VarNames{2} = 'correctAnswer';

export(temp,'file','submission.csv','delimiter',',');

% A = tdfread('sample\_submission.csv','\t');

% for i=1:length(val\_ques\_norm)

%     ACC(i) = (A.id0x2CcorrectAnswer(i,8)==c\_ans(i));

% end

%

% Accuracy = sum(ACC)/length(val\_ques\_norm)

% %%%%%%%%%%%%%%%%%% Ans D done

**CODE FOR string2hash.m**

function hash=string2hash(str,type)

% This function generates a hash value from a text string

%

% hash=string2hash(str,type);

%

% inputs,

%   str : The text string, or array with text strings.

% outputs,

%   hash : The hash value, integer value between 0 and 2^32-1

%   type : Type of has 'djb2' (default) or 'sdbm'

%

% From c-code on : http://www.cse.yorku.ca/~oz/hash.html

%

% djb2

%  this algorithm was first reported by dan bernstein many years ago

%  in comp.lang.c

%

% sdbm

%  this algorithm was created for sdbm (a public-domain reimplementation of

%  ndbm) database library. it was found to do well in scrambling bits,

%  causing better distribution of the keys and fewer splits. it also happens

%  to be a good general hashing function with good distribution.

%

% example,

%

%  hash=string2hash('hello world');

%  disp(hash);

%

% Function is written by D.Kroon University of Twente (June 2010)

% From string to double array

str=double(str);

if(nargin<2), type='djb2'; end

switch(type)

   case 'djb2'

       hash = 5381\*ones(size(str,1),1);

       for i=1:size(str,2),

           hash = mod(hash \* 33 + str(:,i), 2^32-1);

       end

   case 'sdbm'

       hash = zeros(size(str,1),1);

       for i=1:size(str,2),

           hash = mod(hash \* 65599 + str(:,i), 2^32-1);

       end

   otherwise

       error('string\_hash:inputs','unknown type');

end