General Instructions

In this project, you are going to get familiarize with implementing a parallel programming /

distributed computing that includes matrix-vector multiplication, hopefully, in an interesting

context. To put it in a nutshell, you will figure out the list of words which are semantically

most similar to a given input word.

In the subsequent sections, you will be given some background information about the

motivation behind the application, how the semantic meaning of a word is represented and

how the similarity of words is quantified.

Background Information

Motivation: Query Expansion

Suppose that you want to query a keyword in a typical search engine. The search engine

considers not only the query terms that you give as input but also some other terms that

are relevant to your query terms to improve the quality of the result set. As an example,

you may search for the query terms sakura flower usa to look for the places where Japanese

Flower Sakura can be seen in USA (See Figure 1). You may be surprised that the bolded

match terms include not only your query terms but also some additional relevant terms such

as cherry, blossom, tree and matsuri. This method is called query expansion.

The engine is supposed to find the terms that are highly relevant to your query terms.

Considering that the number of distinct terms that is available in a typical large scale search

engine, this process may take too much time. However, this process should be completed

as fast as possible to meet the low latency requirement of the search engines which is to

return results in subseconds [1]. One way to remedy this issue is to distribute the computational

operations into many computers and find the relevant terms with a parallel computing

approach.



Figure 1: Example for a query expansion application in Google.

Word Embeddings

One of the outstanding breakthrough in Natural Language Processing is the introduction of

word embeddings. A word embedding is a fixed-sized vector of real numbers that represents

the position of a specific word in an high dimensional space. Like the stars in our 3D space,

each word is scattered in this high dimensional space. The semantically similar words are

close to each other while semantically different words are apart from each other in this high

dimensional space. For example, the word ’beyaz’ tends to be close to the set of color words

’mavi’, ’sarı ’, ’pembe’ and the word ’telefon’ tends to be in close proximity to the set of

telecommunication words such as ’mobil’, ’gsm’, ’mesaj’, ’internet’ and ’kontör’. In this

project, you are supposed to figure out the set of P relevant words to the given input word in

the given high dimensional space of word embeddings. You may want to refer to the following

video to get more information about the word embeddings (https://bit.ly/2YXCmru).

Cosine Similarity

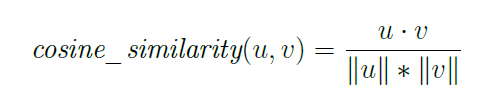
One of the frequently used method to measure the similarity of two vectors is to measure

the angle between these vectors and convert it into a similarity value. Calculating similarity

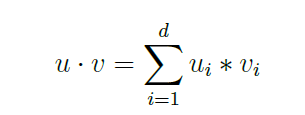
of two vectors by their angle in between is called cosine similarity.

Given two vectors u and v, the cosine similarity of them is calculated as follows:

cosine\_similarity(u, v) =

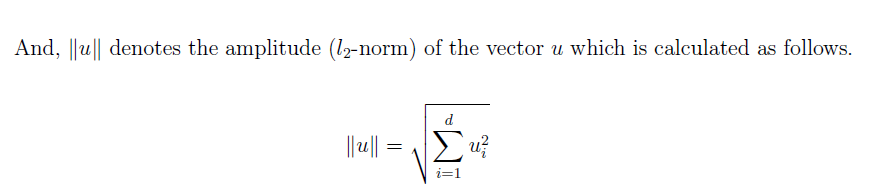


where the operator (·) is the dot product which is calculated as follows:



where ui and vi denotes the value of i’th dimensional unit of the vectors u and v of size

d, respectively.



You may also want to refer to the following video to get a feeling of how dot product

works in the geometrical sense (https://www.youtube.com/watch?v=h0NJK4mEIJU). Note

that if the vectors u and v are unit vectors, that is ||u|| = ||v|| = 1, the cosine similarity

reduces to dot product of two vectors.

Problem Definition

In this problem, you are supposed to implement a query expansion application by using Word embeddings [2, 3, 4]. You are given a matrix which corresponds to the unit word vectors of size 300 for a vocabulary of size 1000. You can download the word embeddings matrix (http://bit.ly/cmpe300-word-embeddings-zip). Each line in the file is tab delimited and each line starts with a word

string that is followed by 300 real numbers. The real numbers in each line corresponds to the

word embeddings of the word given at the beginning of each line. The words are in Turkish.

Given an input word w and the integer value P, you are supposed to distribute the

embedding matrix into P different computational units row-wise (row-major) and return the

list of P similar words to a given input word w. Each computational unit is supposed to look

at its own part of the matrix and to return the most similar word along with the similarity

score to the master node, so, the total number of the similar words will be P in the master

node at the end of the whole operation.

The computational units will be the virtual processors in our MPI implementation. Note

that the computational units would also be fully featured distributed computers without the

loss of the generality in the algorithm.

For example:

Given an input word telefon and P=11, you are supposed to return the following list of

10 words: telefon, cep, iphone, ipad, cihaz, mobil, gsm, mesaj, internet, kontör. Note that

the list includes also the input word.

Please read the query word and number of processors (P+1) as the command line parameters.

Note that the rank of the master node will be zero and the ranks of P slave nodes will

be positive. Then, implement an MPI algorithm which satisfy the following requirements:

• (10 points) Requirement 1: While reading the embedding matrix, distribute it into

the P computational units evenly and row-wise. Note that we assume we do not have

enough memory to read all matrix into a single computational units, so, you should

not read the matrix all at once and distribute later. Instead, you should send parts of

the matrix to the slave units while reading it in the master node.

• (40 points) Requirement 2:After distributing the embedding matrix, the master

node and slave nodes will communicate each other via a command and control protocol. In other words, the master node will send a command, which is an integer, to the slave

nodes and receive their responds to the commands. The commands are listed below:

COMMAND\_EXIT=0: When the user enters EXIT as the query word, the

master node will send this command to shut down all slave nodes and exit (5 points).

COMMAND\_QUERY=1: The master node will send this command to send

the query term to each slave units. When the slave units receive this command, they

are supposed to return the row index of the word if it exists in the part of the embedding

matrix they have, otherwise -1. In addition, the slave node which has the query term

should also send the embedding of the word to the master node. If the query term

doesn’t exist in any slave node, the master node will receive -1 from each slave node

as the word index. In this case, the master node should prompt "Query word was not

found" and start waiting another query term from the user (15 points).

COMMAND\_CALCULATE\_SIMILARITY=2: If themaster node receive

a non-negative word index from a slave node along with the query word embedding

as a responds to COMMAND\_QUERY=1, the master node will send COMMAND\_

CALCULATE\_SIMILARITY=2 along with the query word embedding to

all slave nodes. The slave nodes are supposed to calculate the similarity between the

query word embedding with all embeddings in its own memory. Then, the slave nodes

should return the word along with its embedding which is most similar to the query

word embedding. Note that one of the computational units will return the query word

itself with a similarity value of 1.0. Then, the master node should print the similar

words and their similarity scores which are received from the slave nodes (20 points).

• (20 points) Requirement 3: Implement all the requirements above and let the

user query words in an infinite interactive session just like in a search engine. The

session should end when the user enters EXIT as the query word. You can make

use of Algorithm 1 and Algorithm 2 for master and slave nodes or implement all the

requirements in an other way. After completing the implementation, find P=10 similar

words to each of the following 10 query words in this interactive session. Include your

answers in your report.

boğaziçi:

üniversite:

bilgisayar:

mühendis:

bölüm:

algoritma:

analiz:

ders:

proje:

ödev:

• (30 points) Requirement 4: Write a report of your application. Include all of your

design decisions, assumptions and results, clearly and thoroughly.

• (20 points) BONUS: For each word given in the Requirement 3, if your implementation

lists the top P most relevant words rather than P relevant words, you will

get 20 bonus points!

