Group Name: Athanasios

group's members:

1- Golnoosh Sharifi - 50011414

2- Mahdi Rahimianaraki - 50014390

3- Siarhei Sheludzko - 3092139

4- Aleksei Zhuravlev - 50104961

5- Marcel Melchers - 2897058

1. Orthonormal Vectors:

1. c = ||a|| + ||b||

a and b are orthonormal. It means they are both normal and orthogonal. →

a.b = 0

|| a || = 1

|| b || = 1

→

c = || a || + || b ||

c = 1+1

c = 2

2. d = cosine similarity (a, b)

$\cos(\alpha) = \frac{a.b}{\|a\| \|b\|} = 0$ →

α = 90°

2. Latent Semantic Indexing

2.1:

In matrix factorization we decompose our matrix into two smaller matrices. The first one is term concept matrix, and the other is a document concept matrix.

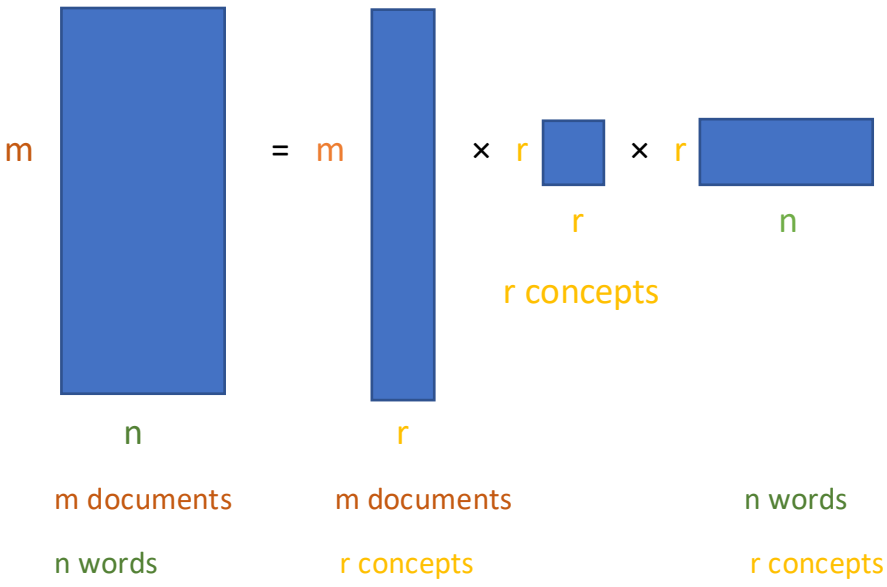
The notion concept is what LSI introduce to us.

The benefit of having these two matrices is that they give us a sophisticated way to do document retrieval.

By SVD we can find concepts in matrices, and also through finding these concepts, we can actually rank the concept and potentially throw away the less important ones. So, we can store our data more efficiency. And also, there is a strong relationship between dimensionality reduction and throwing away information to compression and also recommendation.

SVD decompose our matrix into three smaller matrices.

A [m × n] = U [m × r] S [r × r] $(\mathrm{V}[\mathrm{n} \times \mathrm{r}])^{T}$



In our middle matrix, r is the number of concepts, and this matrix is a diagonal matrix. It means only the diagonal entries are nonzero and also the values along the diagonal, encode the concept strength. They are also ordered.

So, the concept with strongest strength will be at top left and the one with smallest strength will be at the lower right. So that means you have a ranked list of concept strengths going along the diagonal.

2.2:

Our solution link in google colab:

Solution 2.2 colab link

Also, the notebook file of the solution is in the zip file.

3. GloVe

3.1:

GloVe is a method that we can use for training our embedding layer. The main idea of GloVe is modeling the relationships among word triplets.

In GloVe, both “semantical similarities” and “syntactic similarities” are considered. But in LSI only “semantical similarities” between words are captured.

Vector operation within representation is not taken into account in LSI, while it is possible in GloVe.

3.2

$g(w_i, w_j, v_l) = \frac{p(l \mid i)}{p(l \mid j)}$

$g((w_i - w_j)^T v_l) = g(w_i^T v_l - w_j^T v_l) = \frac{p(l \mid i)}{p(l \mid j)}$

Assuming Homomorphism

$$g(w_i^T v_l - w_j^T v_l) = \frac{g(w_i^T v_l)}{g(w_j^T v_l)} = \frac{P(l|i)}{P(l|j)}$$

simplification to allow for modeling the symmetries $C \Leftrightarrow C^T$ and $w_i \Leftrightarrow v_i$

$$g(w_i^T v_l) = P_{ij} = \frac{c_{il}}{c_i}$$

$$\text{if } g(x) = e^x \Rightarrow e^{w_i^T v_l - w_j^T v_l} = \frac{e^{w_i^T v_l}}{e^{w_j^T v_l}}$$

$$\log(e^{w_i^T v_l}) = \log(c_{il}) - \log(c_i)$$

$W_i^T v_l + b_i + q_l = \log(c_{il})$ where the bias terms for i and l are respectively b_i and q_i

4. Recurrent Neural Networks

Exercise 4.1

Compare a vanilla (a.k.a. simple) recurrent neural network (RNN) to a multi-layer perceptron (MLP). What can an RNN model that an MLP is unable to? How does it achieve this feat?

An adapted answer from ChatGPT:

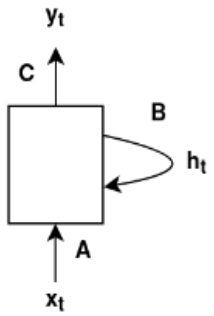


Figure 1: A Elman network

A vanilla recurrent neural network (RNN) aka Elman network is a type of artificial neural network that can process sequential data, such as text, audio, or time series data. In contrast, a multi-layer perceptron (MLP) is a type of feedforward neural network that processes input data in a single pass through the network.

One key advantage of RNNs over MLPs is that they can capture temporal dependencies in data. This means that RNNs can take into account the order or sequence of the input data, whereas an MLP is not able to do this. For example, in the case of processing natural language, an RNN can understand the meaning of a sentence by taking into account the order of the words, whereas an MLP cannot.

RNNs achieve this ability through the use of feedback connections, which allow previous outputs of the network to be fed back into the input of the network. This allows the network to maintain an internal state that can capture information about the past inputs, which can be used to inform the processing of future inputs. Figure 1 shows a Elman network. The internal state at time t (or system state) is denotet with $h_t = f^h(Bh_{t-1} + Ax_t)$ which depends on its previous past system state h_{t-1} and influences the ouput $y_t = f^0(Ch_t)$. y_t is sometimes denoted as x_{t+1} , whether the RNN task is to predict the future or not.

Overall, RNNs are well-suited for modeling sequential data, such as text, audio, or time series data, whereas MLPs are better suited for processing non-sequential data

4.2:

Long Short-Term Memory Networks is an advanced RNN, a sequential network, that allows information to persist. The Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN) that, in certain cases, has advantages over long short-term memory (LSTM). GRU uses less memory and is faster than LSTM, however, LSTM is more accurate when using datasets with longer sequences.

The problem with Recurrent Neural Networks is that they simply store the previous data in their “short-term memory”. Once the memory in it runs out, it simply deletes the longest retained information and replaces it with new data. (RNNs were mostly impractical due to the vanishing and exploding gradient problems during training)

The LSTM and GRU have the same purpose of tracking long-term dependencies effectively while mitigating the vanishing/exploding gradient problems. This is done by LSTMs using input, forget, and output gates, where the input gate controls how much of the new cell state to keep, the forget gate controls how much existing memory to forget, and the output gate controls how much of the cell state is exposed to the next layers. The GRU operates using a reset gate and an update gate. The reset gate sits between the previous activation and the next candidate activation to forget the previous state, and the update gate decides how much of the candidate activation to use in updating the cell state.
LSTMs and GRUs have the ability to keep memory/state from previous activations instead of replacing the entire activation like a vanilla RNN, allowing them to remember features for a long time and allowing backpropagation to happen through multiple bounded nonlinearities, which reduces the likelihood of the vanishing gradient.
LSTMs control the exposure of memory content (cell state) while GRUs expose the entire cell state to other units in the network. The LSTM unit has separate input and forgets gates, while the GRU performs both of these operations together via its reset gate.

5. Revisiting the SMS spam dataset

Our solution link in google colab:

Solution 5 colab link

Also, the notebook file of the solution is in the zip file.

6. Named Entity Recognition with Recurrent Neural Networks

Our solution link in google colab:

Solutions 6 colab link

Also, the notebook file of the solution is in the zip file.