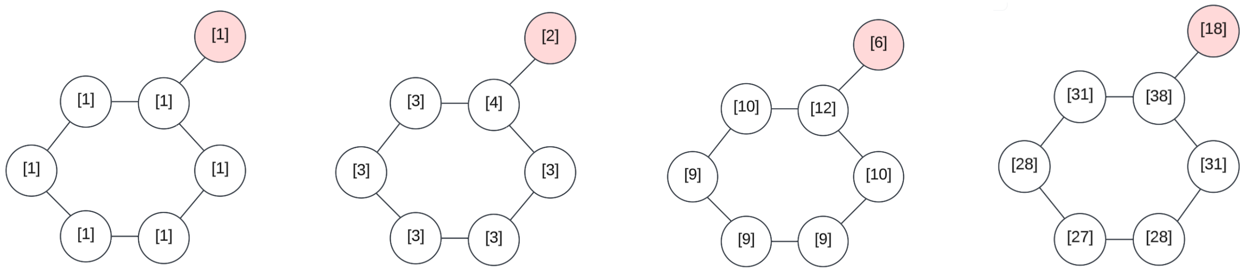
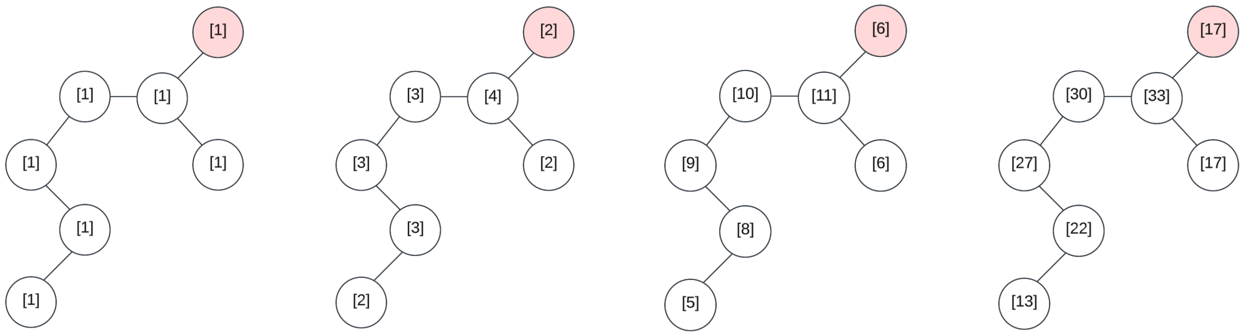
Question 1(a), Homework 4, CS246

In the beggining all nodes have initial feature vector $x = [1]$. In each iteration this vector is updated to the sum of node's current embedding and embeddings of it's neighbors. Lets write the embedding of each node next to it in the picture.

Graph 1:



Graph 2:



We can see that the embeddings for the red nodes in the given graphs begin to differ after three hidden layers. On graph 1 the embedding on 3rd layer of the red node is [18] and on graph 2 it is [17].

To reduce Janossy pooling to mean aggregation, we need to define a function ρ_ϕ such that it computes the mean of the embeddings of the neighbouring nodes of the sequence.

This can be expressed as

$$\rho_\phi(h_{u_1}^{k-1}, h_{u_2}^{k-1}, \dots h_{u_{|N(v)|}}^{k-1})_\pi = \frac{1}{|N(v)|} \sum_{u \in N(v)} h_u^{k-1},$$

here h_u^{k-1} represents the embedding of neighbour node u of v at layer $k-1$ and $N(v)$ represents the neighbourhood of node v. Observe that the right side is not depended on permutation π, therefore the aggregation function which is the average over all permutations from Π can be rewriten as:

$$\begin{aligned}
\text{aggregate}_{\text{janossy}}\left(\left\{h_u^{k-1}, \forall u \in N(v)\right\}\right) &= \frac{1}{|\Pi|} \sum_{\pi \in \Pi} \rho_\phi(h_{u_1}^{k-1}, h_{u_2}^{k-1}, \dots h_{u_{|N(v)|}}^{k-1})_\pi \\
&= \frac{1}{|\Pi|} \sum_{\pi \in \Pi} \frac{1}{|N(v)|} \sum_{u \in N(v)} h_u^{k-1} \\
&= \frac{1}{|\Pi|} |\Pi| \frac{1}{|N(v)|} \sum_{u \in N(v)} h_u^{k-1} \\
&= \frac{1}{|N(v)|} \sum_{u \in N(v)} h_u^{k-1}
\end{aligned}$$

By defining ρ_ϕ in this way, Janossy pooling reduces to mean aggregation, as it computes the mean of the embeddings of all neighbor nodes for each permutation of the input sequence. This aligns with the concept of mean aggregation where information from all neighbors is aggregated by taking the mean of their embeddings.

Janossy pooling is potentially more expressive than mean pooling. This is because Janossy pooling considers all permutations of neighbor embeddings and applies a function (such as an LSTM) to each permutation, whereas mean pooling simply computes the average of neighbor embeddings without considering permutations. We also observed this in our example, since we were able to reduce janossy pooling to mean aggregation function.

Now we consider breadth-first search, where at every step, nodes that are connected to already visited nodes become visited. Message function of this approach is

$$M(h_v^k) = h_v^k.$$

Here the message function returns 1 if the node v was already visited, otherwise it returns 0.

The aggregation function collects messages from all neighboring nodes. We can use the sum of all neighbouring nodes, by which we achive that update function works properly. Define aggregation function as:

$$h_{N(v)}^{k+1} = \sum_{u \in N(v)} h_u^k.$$

With the update function we would like to update the node's representation. Since BFS works in a binary manner (visited or not visited), the update function should set the representation of v to either 0 or 1. It should become 1 if one of the neighboring nodes is 1. That is exactly when an aggregation function is greater than zero:

$$h_v^{k+1} = \left\{ \begin{array}{ll} 1, & h_{N(v)}^{k+1} > 0 \\ 0, & \text{otherwise} \end{array} \right\}.$$

For the aggregation function we could also use logical OR operator. If one of the neighbouring nodes of v is 1, than h_v should be 1 in the next iteration:

$$h_{N(v)}^{k+1} = OR(h_{N(v)}^k).$$

If we defined aggregation function like this, the update function is identical to aggregation function:

$$h_v^{k+1} = h_{N(v)}^{k+1}.$$

When calculating the gain function for wine, we devide all 100 people on those who like wine and those who do not. This way we get two groups of 50 people. Then we further devide each of the two groups on those who like beer and those who do not. In both groups there are 20 people who like beer and 30 who do not. Therefore value of gain function G for wine is 2.0, calculated from

$$\begin{aligned}
G_{wine} &= I(D) - (I(D_L) + I(D_R)) = \\
&= \left(100 \times \left(1 - \left(\frac{50}{100}\right)^2 - \left(\frac{50}{100}\right)^2\right)\right) - \\
&- \left(50 \times \left(1 - \left(\frac{20}{50}\right)^2 - \left(\frac{30}{50}\right)^2\right)\right) - \\
&- \left(50 \times \left(1 - \left(\frac{20}{50}\right)^2 - \left(\frac{30}{50}\right)^2\right)\right) = \\
&= 50.0 - 24.0 - 24.0 = \\
&= 2.0
\end{aligned}$$

When caluclating the gain function for running, we first form two groups, a group of 30 people who like running and a group of 70 who do not. Then we devide a group of 30 who like running to 20 who also like beer and 10 that do not. Similarly we devide the group of 70 to 20 who like beer, the remaining 50 do not. Gain function for running is calculated as follows:

$$\begin{aligned}
G_{running} &= I(D) - (I(D_L) + I(D_R)) = \\
&= \left(100 \times \left(1 - \left(\frac{30}{100}\right)^2 - \left(\frac{70}{100}\right)^2\right)\right) - \\
&- \left(30 \times \left(1 - \left(\frac{20}{30}\right)^2 - \left(\frac{10}{30}\right)^2\right)\right) - \\
&- \left(70 \times \left(1 - \left(\frac{20}{70}\right)^2 - \left(\frac{50}{70}\right)^2\right)\right) = \\
&= 42.0 - 13.333 - 28.571 = \\
&= 0.095.
\end{aligned}$$

We do the same for pizza. First form two groups of people - 80 who like pizza and 20 who dislike it. Then devide the first group on 30 who like beer and 50 who do not, and the second group on 10 who like beer and 10 who do not. Calculating the gain function we obtain:

$$\begin{aligned}
G_{pizza} &= I(D) - (I(D_L) + I(D_R)) = \\
&= 32.0 - 37.5 - 10.0 = \\
&= -15.5.
\end{aligned}$$

We see that gain is maximal for wine. Therefore if we were to maximize the gain G using the gini index metric, we should use wine attribute to split the data at root.

Since attribute a_1 has the most significant dependance on target y, the root node of the decision tree should be assosiated with a_1. This will lead to two child nodes, $a_1 = 0$ and $a_1 = 1$. Other attributes may contribute to splitting decisions, but their impact will be smaller than of a_1.

Because we are building a complete binary tree, we use values of all attributes. Since there are 100 attributes in total and each level of the tree represents a decision based on one of these attributes, the height of the tree, which corresponds to the number of levels, will be equal to the number of attributes.

The desired decision tree should also split on a_1 at the root node, since it is the most informative. This way it will return the correct answer with 99% probability. Splitting the tree further by other attributes would only lower this probability, since the answer is returned by the leaf node and the leaf node has the probability of returning the correct answer lower then 99%.

Lets prove

$$\text{cost}(S,T) \leq 2\sum_{i=1}^{l} \text{cost}(S_i,T_i) + 2\cdot \text{cost}_\omega(\hat{S},T).$$

For every node $x \in S$ we use triangular inequality:

$$d(x,T) \leq d(x,t_{ij}) + d(t_{ij},T),$$

where $t_{ij} \in T_i \subset \hat{S}$ is the centroid nearest to x, meaning $x \in S_i$ for some i. T is defined as in the algorithm.

We can square this inequality, obtaining

$$d(x,T)^2 \leq (d(x,t_{ij}) + d(t_{ij},T))^2 \leq 2d(x,t_{ij})^2 + 2d(t_{ij},T)^2,$$

where in the second inequality we used a fact given in instructions: $(a+b)^2 \leq 2a^2 + 2b^2$.

Because this inequality holds true for every node $x \in S$, it must also hold true for the sum over all $x \in S$, by which we get

$$\sum_{x\in S} d(x,T)^2 \leq 2\sum_{i=1}^{l}\sum_{x\in S_i} d(x,t_{ij})^2 + 2\sum_{t_{ij}\in\hat{S}} \omega(t_{ij})d(t_{ij},T)^2. \tag{1}$$

Lets see that in all three summands we summed over all $x \in S$. To see this in summand on the left is trivial. In the second summand, we sum over all $x \in S_1 \dots x \in S_l$, meaning we sum over all $x \in S_1 \cup \cdots \cup S_l = S$. Here $t_{ij} \in T_i$ is the nearest centroid from $x \in S_i$, defined in the sum. In the third sum, we are summing over all centroids in $\hat{S} = \bigcup_{i=1}^{l} T_i$. Here we count each distance from t_{ij} to T as many times as there are nodes that are nearest to centroid t_{ij}. This is exactly the definition of $\omega(t_{ij})$.

We know that distance from a point to a set of points is the minimum distance from a given point to each point in the set. Since t_{ij} is defined as a centroid from T_i nearest to x, we can rewrite $d(x,t_{ij}) = d(x,T_i)$.

From the definition of cost function we see

$$\sum_{x\in S} d(x,T)^2 = \text{cost}(S,T),$$

$$\sum_{x\in S_i} d(x,T_i)^2 = \text{cost}(S_i,T_i),$$

$$\sum_{t_{ij}\in\hat{S}} \omega(t_{ij})d(t_{ij},T)^2 = \text{cost}(\hat{S},T).$$

Now lets rewrite equation (1). We get the final result

$$\text{cost}(S,T) \leq 2\sum_{i=1}^{l} \text{cost}(S_i,T_i) + 2\cdot \text{cost}_\omega(\hat{S},T).$$

Lets prove

$$\sum_{i=1}^{l} \text{cost}(S_i, T_i) \leq \alpha \cdot \text{cost}(S, T^*).$$

Because we assume we have an algorithm ALG which is an α-approximate k-means clustering algorithm, we know

$$\text{cost}(S_i, T_i) \leq \alpha \min_{|T'|=k} \left\{ \text{cost}(S_i, T') \right\}.$$

Since we take a subset $S_i \subset S$ and a fixed set of centroids T^*, we can bound

$$\min_{|T'|=k} \left\{ \text{cost}(S_i, T') \right\} \leq \text{cost}(S_i, T^*).$$

Lets bound the left side of inequality we are proving

$$\begin{aligned} \sum_{i=1}^{l} \text{cost}(S_i, T_i) &\leq \alpha \sum_{i=1}^{l} \text{cost}(S_i, T^*) \\ &= \alpha \sum_{i=1}^{l} \sum_{x \in S_i} d(x, T^*) \\ &= \alpha \sum_{x \in S} d(x, T^*) \\ &= \alpha \cdot \text{cost}(S, T^*) \end{aligned}$$

In third line we used that $S = S_1 \cup \cdots \cup S_l$ is a disjunct union.

First lets prove

$$\text{cost}_\omega\left(\hat{S}, T\right) \leq \alpha \cdot \text{cost}_\omega\left(\hat{S}, T^*\right). \tag{2}$$

Since we use algorithm ALG on $\hat{S}$ to find T, from the definition of ALG we know

$$\text{cost}_\omega\left(\hat{S}, T\right) \leq \alpha \min_{|T'|=k} \left\{\text{cost}_\omega\left(\hat{S}, T'\right)\right\}$$

Since T^* is fixed set of cardinality k, we know that its cost is at least the minimum cost over all sets of cardinality k. Mathematically this can be written as

$$\min_{|T'|=k} \left\{\text{cost}_\omega\left(\hat{S}, T'\right)\right\} \leq \text{cost}_\omega(\hat{S}, T^*).$$

This proves the first inequality.

Now we would like to prove

$$\text{cost}_\omega(\hat{S}, T^*) \leq 2\sum_{i=1}^{l} \text{cost}(S_i, T_i) + 2 \cdot \text{cost}(S, T^*). \tag{3}$$

For every $x \in S$ we can use triangular ineguality

$$d\left(t_{ij}, T^*\right) \leq d\left(x, t_{ij}\right) + d\left(x, T^*\right),$$

where t_{ij} is centroid nearest to x. Squaring the whole inequality gives

$$d\left(t_{ij}, T^*\right)^2 \leq \left(d\left(x, t_{ij}\right) + d\left(x, T^*\right)\right)^2 \leq 2d\left(x, t_{ij}\right)^2 + 2d\left(x, T^*\right)^2.$$

Since $t_{ij} \in T_i$ is the nearest to x from all point in T_i, we know $d\left(x, t_{ij}\right) = d\left(x, T_i\right)$ from the definition of Euclidean distance between a point and a set. Now summing over all $x \in S$ we obtain

$$\sum_{t_{ij} \in T_i} \omega(t_{ij}) d\left(t_{ij}, T^*\right)^2 \leq 2\sum_{i=1}^{l} \sum_{x \in S_i} d\left(x, T_i\right)^2 + 2\sum_{x \in S} d\left(x, T^*\right)^2.$$

When summing over all $x \in S$ we count every distance $d\left(t_{ij}, T^*\right)^2$ as many times as there are points $x \in S$ that are nearest to t_{ij}, thus we muliply with $\omega(t_{ij})$. Now using the definition of cost function we obtain

$$\text{cost}_\omega(\hat{S}, T^*) \leq 2\sum_{i=1}^{l} \text{cost}(S_i, T_i) + 2 \cdot \text{cost}(S, T^*).$$

With the above proofs the final result is just a simple calculation:

$$\begin{aligned}
\text{cost}\,(S,T) &\leq 2\cdot \text{cost}_\omega(\hat{S},T) + 2\sum_{i=1}^{l}\text{cost}(S_i,T_i) && (4)\\
&\leq 2\alpha\cdot\text{cost}_\omega\left(\hat{S},T^*\right) + 2\sum_{i=1}^{l}\text{cost}(S_i,T_i) && (5)\\
&\leq 2\alpha\cdot\text{cost}_\omega\left(\hat{S},T^*\right) + 2\alpha\cdot\text{cost}(S,T^*) && (6)\\
&\leq 2\alpha\left(2\sum_{i=1}^{l}\text{cost}(S_i,T_i) + 2\cdot\text{cost}(S,T^*)\right) + 2\alpha\cdot\text{cost}(S,T^*) && (7)\\
&\leq 2\alpha\left(2\alpha\cdot\text{cost}(S,T^*) + 2\cdot\text{cost}(S,T^*)\right) + 2\alpha\cdot\text{cost}(S,T^*) && (8)\\
&\leq \left(4\alpha^2+6\alpha\right)\cdot\text{cost}(S,T^*), && (9)
\end{aligned}$$

where we utilized part 3(a) to address inequality (4). Then, in (5), we referenced equation (2), followed by employing part 3(b) in (6). Inequality (7) relied on (3). In (8), we once more employed part 3(b), and finally, in (9), we utilized basic multiplication and summation to derive the final result.

In this question we would like to prove

$$\Pr\left[\tilde{F}[i] \leq F[i] + \epsilon t\right] \geq 1 - \delta. \tag{10}$$

Rewrite this inequality as

$$1 - \Pr\left[\tilde{F}[i] \geq F[i] + \epsilon t\right] \geq 1 - \delta,$$
$$\Pr\left[\tilde{F}[i] \geq F[i] + \epsilon t\right] \leq \delta.$$

Therefore proving (10) is equal to proving the last inequality above.

Lets rewrite the left side. First we use the definition of $\tilde{F}[i]$.

$$\Pr\left[\tilde{F}[i] \geq F[i] + \epsilon t\right] = \Pr\left[\min\{c_{j,h_j(i)}\} > F[i] + \epsilon t\right]$$

Since all $c_{j,h_j(i)}$ are greater then their minimum, this probability is equal to the following.

$$= \Pr\left[\bigwedge_{j=1}^{\lceil \log \frac{1}{\delta} \rceil} \left(c_{j,h_j(i)} > F[i] + \epsilon t\right)\right]$$

Because all expressions in the conjunction are independent, we can rewrite the latest probability as a product.

$$= \prod_{j=1}^{\lceil \log \frac{1}{\delta} \rceil} \Pr\left[c_{j,h_j(i)} > F[i] + \epsilon t\right] = \prod_{j=1}^{\lceil \log \frac{1}{\delta} \rceil} \Pr\left[c_{j,h_j(i)} - F[i] > \epsilon t\right]$$

Now we can use Markov inequality on each factor, since both sides are positive. This follows from $F[i] \leq \tilde{F}[i] \leq c_{j,h_j(i)}$ for every j and i.

$$\leq \prod_{j=1}^{\lceil \log \frac{1}{\delta} \rceil} \frac{\mathbb{E}[c_{j,h_j(i)} - F[i]]}{\epsilon t} = \prod_{j=1}^{\lceil \log \frac{1}{\delta} \rceil} \frac{\mathbb{E}[c_{j,h_j(i)}] - F[i]}{\epsilon t}$$

From instructions we know that for any $1 \leq i \leq n$ and $1 \leq j \leq \lceil \log \frac{1}{\delta} \rceil$ it holds $\mathbb{E}[c_{j,h_j(i)}] \leq F[i] + \frac{\epsilon}{e}(t - F[i])$, therefore we can bound the last expression with

$$\leq \prod_{j=1}^{\lceil \log \frac{1}{\delta} \rceil} \frac{F[i] + \frac{\epsilon}{e}(t - F[i]) - F[i]}{\epsilon t} = \prod_{j=1}^{\lceil \log \frac{1}{\delta} \rceil} \frac{(t - F[i])}{et}$$

We see that the above expression is not dependent on j, therefore it is equal to

$$= \left(\frac{(t - F[i])}{et}\right)^{\lceil \log \frac{1}{\delta} \rceil} = \left(\frac{1}{e}\left(1 - \frac{F[i]}{t}\right)\right)^{\log \frac{1}{\delta}}$$

$F[i]$ is defined as the number of times i appeared in the data stram S, therefore it is bounded by $0 \leq F[i] \leq t$. Dividing this inequality with t we get $0 \leq \frac{F[i]}{t} \leq 1$.

$$\leq \left(\frac{1}{e}\right)^{\log \frac{1}{\delta}} = e^{\log \delta} = \delta$$

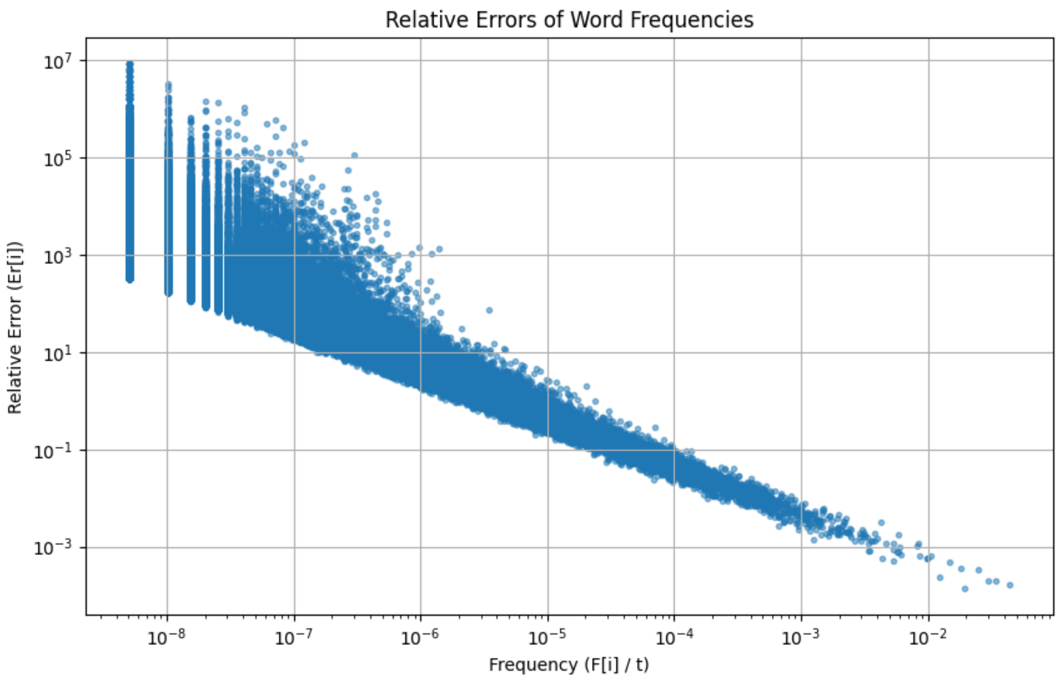


Figure 1: The plot of relative errors depending on word frequency.

What is an approximate condition on a word frequency in the document to have a relative error below $1 = 10^0$?

The frequency of such words should be approximately higher than 10^{-5}.

Information sheet
CS246: Mining Massive Data Sets

Assignment Submission Fill in and include this information sheet with each of your assignments. This page should be the last page of your submission. Assignments are due at 11:59pm and are always due on a Thursday. All students (SCPD and non-SCPD) must submit their homework via Gradescope (http://www.gradescope.com). Students can typeset or scan their homework. Make sure that you answer each (sub-)question on a separate page. That is, one answer per page regardless of the answer length. Students also need to upload their code on Gradescope. Put all the code for a single question into a single file and upload it.

Late Homework Policy Each student will have a total of *two* late periods. *Homework are due on Thursdays at 11:59pm PT and one late period expires on the following Monday at 11:59pm PT*. Only one late period may be used for an assignment. Any homework received after 11:59pm PT on the Monday following the homework due date will receive no credit. Once these late periods are exhausted, any assignments turned in late will receive no credit.

Honor Code We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down their solutions independently, i.e., each student must understand the solution well enough in order to reconstruct it by him/herself. Students should clearly mention the names of all the other students who were part of their discussion group. Using code or solutions obtained from the web (GitHub/Google/previous year's solutions etc.) is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code very seriously and expect students to do the same.

Your name: Lucija Fekonja

Email: lf90992@student.uni-lj.si **SUID:** 27232071

Discussion Group: Nik Mrhar

I acknowledge and accept the Honor Code.

(Signed) LF