1 Question 1

It is obvious that the density increases by increasing the window size since we will capture more couples of words and so more edges.
But the fact that the density never reaches 1 even when we use a window that includes all the terms in the document is not true. Since there are texts where we can capture all the combinations of the words. For example if we want to have a density of 1 with window size of 4 we can just use 4 words that will exist after cleaning the text and then add to it the first 3 words Example

'graph text mining density graph text mining'

we will get a dinsity of 1 with window = 4 since all combinations are captured Fig. 1

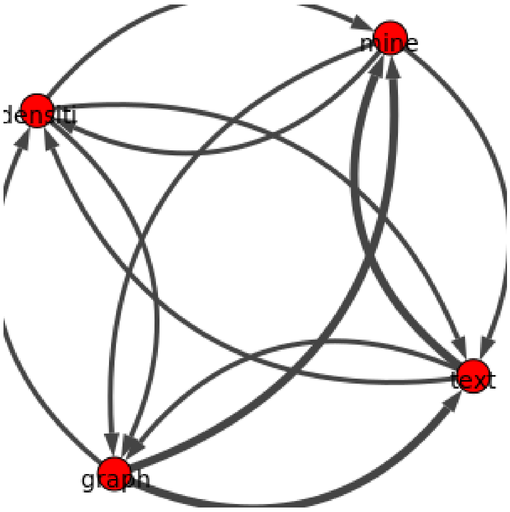

Figure 1: This is a caption.

2 Question 2

The time complexity of k core decomposition:
$n = |V|$
m= $|E|$
neigh$_m$ = mean number of neighbors of the nodes.

Line 1 : Calculate the degrees of each node $O(max(m, n))$ which is less then n^2
Line2: while $O(n)$: Loop over all the vertices
Line 3: $O(n^2)$: Get the vertex with the lowest weight, but since the number of vertices will decrease by 1 at each iteration so the number of operation considering the loop will be

$$\sum_{i=1}^{n} i = O(n^2)$$

line 6 and 7 : $O({neigh_m}^2)$: To delete the edges of a node from an adjacency list we need to loop over all the edges. But the number of edges will decrease at each time but we can consider an upper bound m but in the average case scenario it is the mean of the number of neighbors of all the nodes squared.

So the total complexity is

$$O(n^2 + n * {neigh_m}^2)$$

Or an upper bounded complexity of

$$O(n^2 + n * m)$$

3 Question 3

The performance of k-core and weighted k-core algorithm compared to PageRank and TFIDF with getting top 0.33 of the ranked words i. Overall, PageRank and TFIDF have similar results, with a precision higher than the recall. It is the opposite of the unweighted k-core. The k-core gives better F-1 score which is the most important since it tends to extract a main core with a lot of vertices since the k-core condition can be interpreted as a set of keywords that co-occur with at least k other keywords, so the recall would be higher as shown by Fig 2 .

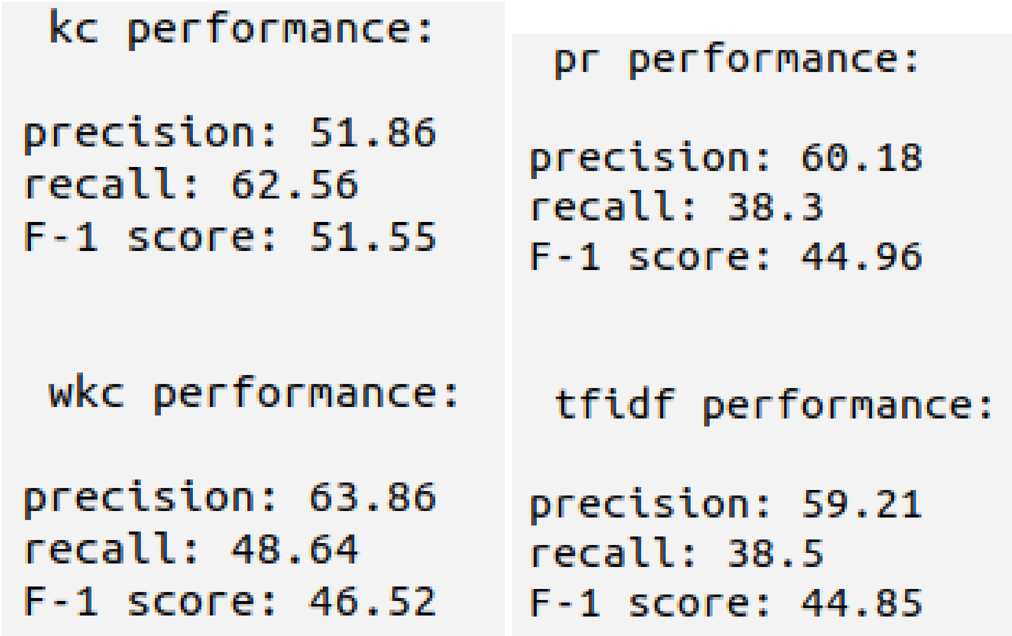

Figure 2: This is a caption.

4 Question 4

The main advantages of k-core is that is able to capture term dependence and term order via directed graph edges.
Another advantage of weighted k-core is that it encodes the strength of the dependence as edge weights.

On the other hand K-core decomposition suffers from the following limitations: (1) k-core is good but not best in capturing cohesiveness; (2) retaining only the main core (or truss) is suboptimal, as one cannot expect all the gold standard keywords to be found within a unique subgraph actually, many valuable keywords live in lower levels of the hierarchy. [2]

Also one of the drawbacks of the k-core decomposition is that the nested k-core subgraphs do not satisfy a natural density property, simply defined as the ratio between the number of edges and nodes of the subgraph. In other words, the maximal k-core subgraph is not necessarily the densest subgraph of the graph. Since it is a greedy algorithm, we may have a node (word) with a high degree which is connected to low degree nodes. So when we remove iteratively low degree nodes, the high degree one will loose its importance[1]

5 Question 5

To improve k-core decomposition we can consider words from the other k-1 cores if they satisfy a threshold of density of the subgraphs. Or we can add backward edges to capture the dependence of words in bidirectional way.

References

[1] Apostolos Papadopoulos Michalis Vazirgiannis Fragkiskos Malliaros, Christos Giatsidis. The core decomposition of networks: Theory,algorithms and applications. 2019.

[2] Rousseau and Vazirgiannis. Main core retention on graph-of-words for single-document keyword extraction. 2015.