ABBOUDI Mohammed Amine
mohammed-amine.abboudi@polytechnique.edu

Lab session # 1
ALTEGRAD 2019

10/13/19

1 Question 1

Given that the density is defined as : $d = \frac{|E|}{|V||V-1|}$ and the number of nodes is only dependent on the document and not the window size, it naturally follows that the density would increase along with the window size.
A we can see in figure 1, augmenting the window size increases the number of edges between nodes (as it allows more distant words to be associated) and hence augmenting the density.

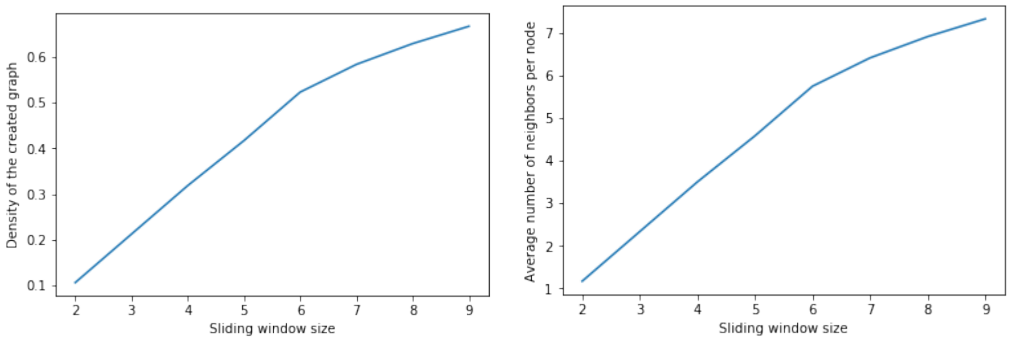

Figure 1: Plots of the window size in regards to graph density (left) and average number of edges per node (right).

2 Question 2: Calculation of the complexity of the naive *unweighted* *k*-core algorithm

We make the assumption that the Graph G is generated beforehand and is defined by a its adjacency list.

Step 1: visits each edge exactly once in order to calculate the degrees of each node. So its complexity is $\mathcal{O}(|E|)$.

Steps 2 & 3: The main loop, which visits each node v exactly once, starts with a minimum extracting function, which goes over the the dictionary p once, but since it is updated each iteration, specifically in regards to size (we get rid of the node with smallest value of p) the complexity of the sorting is $\mathcal{O}(\frac{|V|.(|V|-1)}{2}) = \mathcal{O}(|V|^2)$.

Step 4: is a simple variable allocation, i.e. $\mathcal{O}(1)$, which is looped over each v. It is therefore $\mathcal{O}(|V|)$.

Step 5: We execute a function which returns the neighbors of v that are on average N, N is a variable we introduce that denotes the average number of neighbors of a node in $\mathcal{G}$, it becomes a step of complexity $\mathcal{O}(N \times |V| = \mathcal{O}(|E|))$.

Steps 6 & 7: The algorithm then deletes the node ($\mathcal{O}(1)$) and all its edges, which are averaged as N since we do not want to assume the worst case scenario at each iteration which would require the deletion of $|V|-1$ edges in the case of an un-directed graph and $2(|V|-1)$ in the case of a directed one. The complexity for the deletion of edges is $\mathcal{O}(N \times |V|)$.

Step 8: The algorithm then performs a process of complexity one for all neighbors of v which are again at most $|V|-1$. On average the complexity of such a loop is $\mathcal{O}(N)$.

To conclude, each iteration of the algorithm has a complexity of $\mathcal{O}(|V|+N+N+N) = \mathcal{O}(|V|+N)$, and since each node is visited exactly once, the total complexity of the naive *unweighted* *k*-core algorithm is

$$\mathcal{O}(|V| \times (|V|+N))$$

We choose not simplify the expression any further, but one can choose to suppose that since at most an un-directed graph can have at most $|V| \times (|V| - 1)$ edges which is of the order of $|V|^2$, the complexity becomes $\mathcal{O}(|V|^2)$.

3 Question 3

The performance of the keyword extraction methods is very comparable to the baseline, i.e. PageRank with regards to their respective precision. However, when it comes to recall, the unweighted k-core algorithm almost doubles that of PageRank and TF-IDF. But its precision drops significantly, making it the lowest of all the explored methods. We notice the emergence of a trade-off.

Extraction method	Precision	Recall	F1-score
Weighted k-core	63.86	48.64	46.52
Unweighted k-core	51.86	62.56	51.55
PageRank	60.18	38.3	44.96
TF-IDF	59.21	38.5	44.85

Table 1: A summary of the performance of the different approaches.

4 Question 4

4.1 Advantages and disadvantages of the methods

4.1.1 Unweighted k-core

The unweighted $k-$core algorithm achieves a relatively high recall score (62.56%) which means that it generates the most complete results. But its results are the least pure, i.e. have the lowest precision.

4.1.2 Weighted k-core

On the other hand, the weighted $k-$core algorithm is more conservative when it comes to its threshold of inclusivity, and manages to score the second highest recall score.

The unweighted harmonic mean of the two scores gives a slight advantage to the unweighted $k-$core algorithm even though it performed poorly on the precision metric. However one could modify this score by calculating a weighted F-score, which gives more importance to a given metric depending on the use-case of the algorithm.

5 Question 5

By running the algorithms with slightly modified parameters, more importantly the window size we can notice that precision is positively affected by a decrease in window width, while recall is negatively impacted by it. The opposite is true when window size is increased.
We can therefore say that given what metric is most critical, the score can be ameliorated with an appropriate change to the window size w.
We could also alter the keyword selection mechanism, for example instead of choosing the nodes that belong to the main core sub-graph, one could choose the nodes whom at least one of its edges belongs to at least $K_{max} - 2$ triangles, where K_{max} refers to its main truss.[1] This technique could aid in filtering out noisy and less-cohesive elements of the former sub-graph and could help improve its precision score.

References

[1] Cohen J. Attention is all you need. In *Trusses: Cohesive Subgraphs for Social Network Analysis*, 2008.