1 Question 1

Window Size	2	3	4	5	6	7	8	9
Density	0.106	0.212	0.318	0.417	0.523	0.583	0.629	0.667

Table 1: Influence of the window size on density

The higher the widow size, the higher the density. The density stays lower than one though.

2 Question 2

Algorithm 1 k-core decomposition

Input: graph $G = (V, E)$
Output: dict of core numbers c

1: $p \leftarrow \{v : degree(v)\} \ \forall v \in V$
2: **while** $|V| > 0$ **do**
3: $v \leftarrow$ element of p with lowest value
4: $c[v] \leftarrow p[v]$
5: neighbors $\leftarrow \mathcal{N}(v, V)$
6: $V \leftarrow V \setminus \{v\}$
7: $E \leftarrow E \setminus \{(u, v) | u \in V\}$
8: **for** $u \in$ neighbors **do**
9: $p[u] \leftarrow \max(c[v], degree(u))$
10: **end for**
11: **end while**

Figure 1: k-core Algorithm

Line per line, the complexities are the following :

1. O($|V|$) considering that the degree is obtained in O(1)
2. Loop of length $|V|$
3. O($|V|$)
4. O(1)
5. O(1) considering the adjacency list representation
6. O(1)
7. O($|V|$)
8. Loop of length ($|V|$) at worse (when the graph is fully connected)
9. O(1)

The complexity of the naive version of our algorithm is $0(|V|^2)$. In practice here, the number of neighbours is limited by the window size, and the densities previously shown illustrate the complexity reduction.

3 Question 3

Both k-core and weighted k-core outperform PageRank and Tf-Idf. Note that k-core has more false positives but less false negatives than weighted k-core, and performs better overall.
Also, the fact that graph based method outperform Tf-idf proves the interest of the graph approach.

	Precision	Recall	F1-Score
k-core	51.86	**62.56**	**51.55**
Weighted k-core	**63.86**	48.64	46.52
PageRank	60.18	38.3	44.96
Tf-idf	59.21	38.5	44.85

Table 2: Performances of the different algorithms (window size = 4)

4 Question 4

On the performance side, as stated in the previous question, the advantage of k-core is that it produces fewer false negatives, and the advantages of weighted k-core is that it produces fewer false positives. Overall, one might indeed prefer to extract fewer keywords, with a higher correctness ratio, therefore weighted k-core may be more adapted. However, weighted k-core's complexity is higher than k-core's one.

5 Question 5

On the one hand, there is still a margin of improvement considering preprocessing. Many transformations can be explored and could potentially help the performances. For instance, using lemmatization in addition to or instead of stemming. Or using a dictionary synonym to replace repeating keywords.
On the other hand, these approaches only consider unigrams which is restricting in keywords extraction. A e.g the trigram "*bag-of-words*" can very well be a significant keyword in a NLP paper, whereas the unigram "*bag*" is most likely useless and "*of*" will be removed as stopword.