Del Gallo Adriano
delgallo.adriano@gmail.com

Lab session # 1
ALTEGRAD 2019

10/13/19

1 Question 1

We see from our testing that the density is a non decreasing function of the parameter w. Indeed by increasing the size of the window, we increase the possibility of co-occurrences between words in a single window opening the possibility of forming more links, thus increasing $|E|$ Working with a small window might prevent creating superficial connection between words but may also impact how much context is taken into account. For our keyword extraction a compromise must be found, we should select the window size which maximises the scores of our algorithm.

2 Question 2

Algorithm 1 Complexity of naïve k-core

$O(\|V\|)$	▷ The degree is the length of each list
$O(\|V\|)$	▷ $\|V\|$ times in the loop
$O(\|V\|)$	▷ Minimum search in a list
$O(1)$	
$O(1)$	▷ access one line of adjacency list A
$O(1)$	
$O(\|V\|)$	▷ We delete the edges by going through one column of A
$O(U)$	▷ U is the average number of neighbors
$O(U)$	

Putting everything together, this gives a complexity of $O(|V| + |V|(|V| + 1 + 1 + 1 + |V| + U^2))$
This simplifies into $O(|V|(|V| + U^2))$.

3 Question 3

Score	K-Core	WK-Core	PageRank	TFIDF
Precision	51.86	63.86	60.18	59.21
Recall	62.56	48.64	38.30	38.50
F1-Score	51.55	46.52	44.96	44.85

Table 1: Scores for Hulth2003 data set

The K-Core approach performs the best in recall but not so much on precision. On the other hand the Weighted K-Core algorithm has the highest precision but does not score very well on recall.
TFIDF and Page Rank give similar results which are a good precision (although lower than with WKCore) but have a low recall. They are very task specific
The F1 score summarizes these tendencies into one. We see that according to this score, k-core is the best algorithm overall. The task specific ones have a poor overall score.

4 Question 4

4.1 K-Core

The K-Core approach performs the best in the recall category, as it proposes a high number of candidates for keywords but it also the reason it has a lower precision as the high number of false positives impacts the pertinence of a keyword.

4.2 Weighted K-Core

Adding weights to the K-Core approach increases the variability in the degrees compared to the unweighted one. As a consequence, the final core contains fewer candidates and may miss some of the keyword (which is observed by the drop of recall performance) but the selection is very pertinent as their importance was increased by the weights. This method could be very useful for tasks such as labelling.

Both these graph based approaches have the advantage of being overall more efficient than the widespread tfidf and pagerank, while also having a solution in linear time. However, we cannot anticipate how many keywords can be extracted.

5 Question 5

To improve those approaches, we could either add or remove keywords from the final list. In the case of k-core, the main problem being that there are too many keywords extracted, one could keep the words that maximise a more general property of the graph (betweennes, flow, etc), not just individual ones as before, and delete the others.
The same thing can be done with the weighted k-core approach, where we would instead select keywords from an outside core according to a certain criteria. An example of this is Core Rank.
We could also try to incorporate prior knowledge of a dictionary for the set of documents and apply a multiplying factor to the weights to increase some keywords' importance.