Keywords extraction using GRAPH DATA

In this project, we will discover how methods from social network analysis can be applied to word co-occurrence networks to **extract keywords**. Keyword extraction is a fundamental NLP task used in many areas like information retrieval (search engines), summarization, natural language generation, visualization... Today, we will focus on **unsupervised single-document keyword extraction**.

**Notation**: in what follows,G(V; E)is a graph with **|**V**|** nodes (or vertices) and|E|edges (or links).

N(v; U) designates the immediate neighbors of v in U ⊂ V .

**Igraph**: the nodes and edges of a graphgcan be accessed collectively or individually (throughindexing), along with their attributes, such as ’name’ or ’weight’, via, e.g., g.vs[’name’] or g.es[index][’weight’]. Documentation of all methods and functions can be found at [http:](http://igraph.org/python/doc/igraph.GraphBase-class.html) [//igraph.org/python/doc/igraph.GraphBase-class.html](http://igraph.org/python/doc/igraph.GraphBase-class.html).

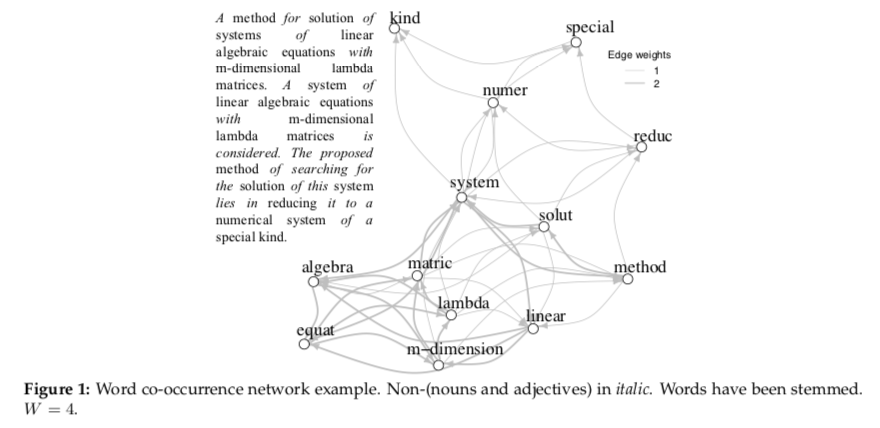
1. **Text preprocessing**

Before constructing a word co-occurrence network, the document needs to be cleaned. The standard steps include (1) conversion to lower case, (2) punctuation removal, (3) tokenization, and (4) stopword removal. Additionally, for keyword extraction, (5) part-of-speech-based filtering (e.g., retaining only nouns and adjectives) and (6) stemming (“winner”, “winning”, and “win” ! “win”) can be useful.

These steps are implemented in the clean text simple function, found within the library.py file.

1. **Word co-occurrence networks**

There are many ways to represent text as a graph. Today, we will use the classical statistical approach of [[6],](#page6) based on the distributional hypothesis (“We shall know a word by the company it keeps” [[3])](#page6). This method applies a fixed-size sliding window of size W over the text from start to finish. Each unique term in the document is represented by a node of the graph, and two nodes are linked by an edge if the terms they represent co-occur within the window. Edge weights are co-occurrence counts. Unlike the vector space model that assumes term independence, this representation captures term dependency, and even term order, if directed edges are used (see Fig. [1)](#page2).



**Figure 1:** Word co-occurrence network example. Non-(nouns and adjectives) initalic. Words have been stemmed.

* = 4.

1. **Keyword Extraction**

**3.1 Influential words**

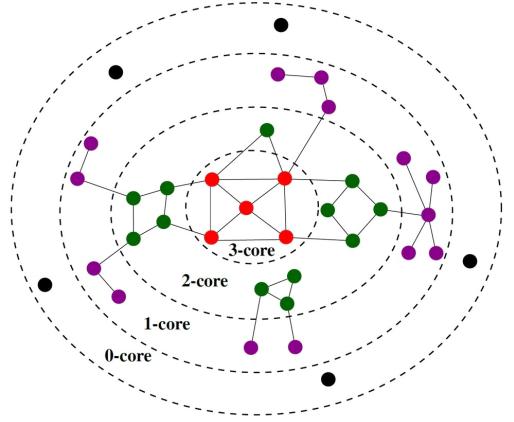
In social networks, it was shown that **influential spreaders**, that is, those individuals that can reach the largest part of the network in a given number of steps, are better identified via their **core numbers** rather than through their PageRank scores, betweenness centralities, or degrees [[5].](#page6) For instance, a less connected person who is strategically placed in the core of a network will be able to disseminate more than a hub located at the periphery of the network, as shown in Fig. [2.](#page3)



Interestingly, taking into account the cohesiveness information captured by graph degeneracy was shown to vastly improve keyword extraction performance [[7,](#page6) 9], meaning that natural language features an important” social” aspect. Keywords can thus be seen as “influential” words.

**3.2** **Graph degeneracy**

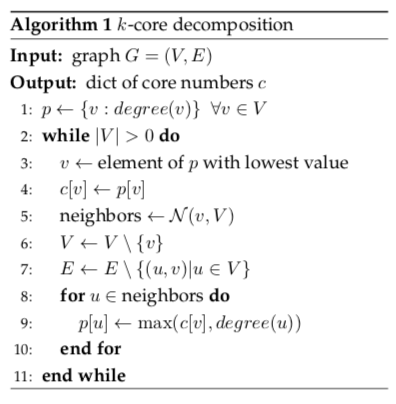
The concept of graph degeneracy was introduced by [[8]](#page6) with the k-core decomposition technique and was first applied to the study of cohesion in social networks.

**k-core** A core of orderk(ork-core) ofGis a maximalconnected subgraph of G in which every vertex v has at least degree k (i.e., k neighbors).

**k-core decomposition** As shown in Fig. [3,](#page3) the k-core decomposition of G is the set of all its cores from 0 (G itself) to kmax (its main core). It forms a hierarchy of nested subgraphs whose cohesiveness and size respec-tively increase and decrease with k. The **core number** of a node is the highest order of a k-core subgraph that contains this node. The main core of G is a coarse ap-proximation of its densest subgraph.

**Figure 3:** Unweightedk-core.

Algorithm [1](#page4) shows the unweighted k-core al-gorithm. It involves a pruning process that removes the lowest degree node at each step, where the degree of a node is simply its num-ber of immediate neighbors. By using the sum of the weights of the incident edges as the de-gree, we obtain the weighted k-core algorithm. The unweighted and weighted k-core algo-rithms can be implemented with very af-fordable time complexities O(|E|) [[2]](#page6) and O(|E| log |V|) [[1],](#page6) by using specific strategies and data structures.



1. **Keyword extraction**

**4.1** **Data set**

We will use the test set of the Hulth 2003 dataset [[4],](#page6) that you can find inside the datanHulth2003testingn folder. This dataset contains 500 scientific paper abstracts. For each abstract in the (abstractsn folder), human annotators have provided a set of keywords (uncontrn folder), that we will consider as ground truth. The keywords were freely chosen by the annotators and some of them may not appear in the orig-inal text. Thus, getting a perfect score is impossible on this dataset.

**4.2** **Baselines**

We will evaluate the performance of the k-core-based approach against that of PageRank (applied on the same graphs) and the vector space representation with TF-IDF coefficients. For each baseline, the top p = 33% percent nodes will be retained as keywords.



**Figure 4:** The main core of the graph can be used as the keywords for the document.

**Assumption**: both fork-core and weightedk-core, we will extract themain coreof the graph as key-words.

**4.3** **Performance evaluation**

We will evaluate the performance of the different techniques in terms of macro-averaged precision, recall and F1 score. Precision can be seen as the purity of retrieval while recall measures the completeness of retrieval. Finally, the F-1 score is the harmonic mean of precision and recall. More precisely, these metrics are defined as follows:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| precision = |  | tp |  | recall = | tp |  | F1-score = 2 | Precision.recall |
| tp + fp | | | tp + fn | | precision + recall |
|  |  |  |

where tp, fp and fn respectively denote the number of true positives (the keywords returned by the system which are also part of the ground truth), false positives (the keywords returned by the system which are not part of the ground truth), and false negatives (the keywords from the ground truth that are not returned by the system). Finally, macro-averaging consists in computing the metrics for each document and calculating the means at the collection level.