A web application for Graph-of-Words-based text visualization and summarization

Antoine J.-P. Tixier, Konstantinos Skianis, and Michalis Vazirgiannis
Computer Science Laboratory
École Polytechnique, France

Abstract

We introduce `GoWvis`[1], an interactive web application that represents any piece of text inputted by the user as a graph-of-words and leverages graph degeneracy and community detection to generate an extractive summary (keyphrases and paragraph) of the inputted text in an unsupervised fashion. The entire analysis can be fully customized via the tuning of many text preprocessing, graph building, and graph mining parameters. Our system is thus well suited to educational purposes, exploration and early research experiments. The new summarization strategy we propose also shows promise.

1 Introduction

The term independence assumption made by the traditional Bag-of-Words (BoW) representation of text comes with many limitations. One approach that challenges this assumption is the Graph-of-Words model (GoW). As shown in Figure 1, it represents a textual document as a graph whose vertices are unique terms in the document and whose edges capture term co-occurrence within a window of predetermined, fixed size, that is slided over the entire document from start to finish.

This approach is statistical, as terms are linked based on local context of co-occurrence only, regardless of any semantic or syntactic information. Unlike the well-known BoW representation, GoW encodes term dependency and term order (via edge direction). The strength of the dependence between two words can also be captured by assigning a weight to the edge that links them. While other definitions can be used, we will consider in

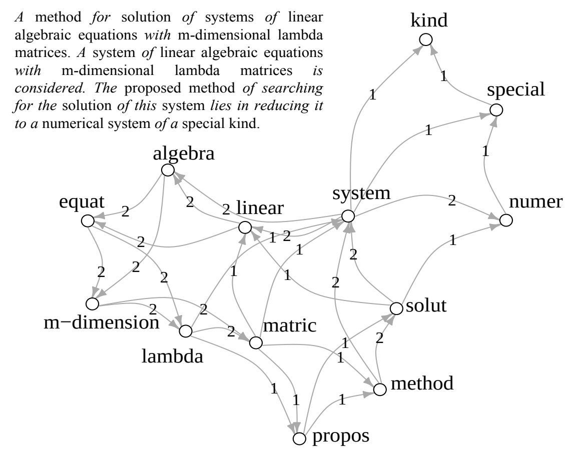
A method *for* solution *of* systems *of* linear algebraic equations *with* m-dimensional lambda matrices. *A* system *of* linear algebraic equations *with* m-dimensional lambda matrices *is considered. The* proposed method *of searching for the* solution *of this* system *lies in reducing it to a* numerical system *of a* special kind.

Figure 1: Graph-of-Words representation

this paper edge weights to be integers matching co-occurrence counts.

GoW can be tracked back to the works of (Mihalcea and Tarau, 2004) and (Erkan and Radev, 2004) who applied it to the tasks of unsupervised keyword extraction and extractive single document summarization. Notably, the former effort ranked nodes based on a modified version of the PageRank algorithm.

Recently, (Rousseau and Vazirgiannis, 2015) showed that degeneracy-based approaches (i.e., extracting dense, cohesive subgraphs) outperform PageRank for unsupervised keyword extraction. We will show in subsection 3.4 how combining this strategy with graph clustering may improve summarization performance for multitopic documents. Other NLP tasks on which GoW-based approaches have reached new state-of-the-art include ad-hoc information retrieval (Rousseau and Vazirgiannis, 2013) and document classification (Rousseau et al., 2015; Malliaros and Skianis, 2015). The high success, promising potential and visual nature of the GoW representation was the impetus for the development of `GoWvis`.

[1]`https://safetyapp.shinyapps.io/GoWvis/`

The remainder of this paper is organized as follows: Section 2 provides some background on graph degeneracy and community detection, Section 3 presents our system, and finally, Section 4 concludes and discusses future work.

2 Graph mining

2.1 Graph degeneracy

k-core. A core of order k (or k-core) of a graph G is a maximal connected subgraph of G in which every vertex v has at least degree k (Seidman, 1983). It is a relaxation of a clique: a k-core with $k + 1$ members is a subgraph where every two nodes are adjacent, that is, a clique (Luce and Perry, 1949). In the classical unweighted case, edge weights are not taken into account and thus the degree of a node v is simply equal to the number of its neighbors. In the weighted (or generalized) case, the degree of a vertex v is the sum of the weights of its incident edges.

k-core decomposition. The k-core decomposition of a graph G is the list of all its cores from 0 (G itself) to k_{max} (its main core). It forms a hierarchy of levels that are recursively included in one another and whose cohesiveness and size respectively increases and decreases with k (Seidman, 1983). A linear (resp. linearithmic) time algorithm for k-core decomposition can be found in (Batagelj and Zaveršnik, 2002) for the unweighted (resp. weighted) case. Both algorithms implement a pruning process that removes the lowest degree node at each step.

The **core number** of a node is the highest order of a core that contains this node. Nodes with high core numbers have the desirable property of not only being central (like nodes with high degree centrality) but also part of cohesive subgraphs with other central nodes (i.e., the other members of the upper cores). For this reason, they make, among other things, influential spreaders (Kitsak et al., 2010) and good keywords (Rousseau and Vazirgiannis, 2015).

The main core of a graph yields a fast (but rough) approximation of its densest subgraph. Indeed, the main core still can contain a large portion of the nodes in the graph. As (Seidman, 1983) puts it, k-cores should be regarded as *seedbeds* within which it is possible to find more cohesive subgraphs.

k-truss. A relatively new triangle-based extension of k-core that yields densest subgraphs is k-truss (Cohen, 2008). More precisely, the k-truss of a graph G is the largest subgraph of G in which every edge belongs to at least $k - 2$ cycle subgraphs of length 3 (i.e., triangles). Put differently, every edge in the k-truss joins two vertices that have at least $k - 2$ common neighbors.

k-truss decomposition. The k-truss decomposition of a graph G is the set of all its k-trusses from $k - 2$ to k_{max}. The k-trusses corresponds to densely connected *subsets* of the k-cores, i.e., their essential parts (Malliaros et al., 2016). The maximal k-truss thus yields a smaller and denser subgraph of G that better approximates its densest subgraph. Nevertheless, the finer resolution of the k-truss decomposition comes at the cost of a greater complexity, polynomial in the number of edges (Wang and Cheng, 2012).

By analogy with k-core, the **truss number** of an *edge* is the highest order of a truss the edge belongs to. By extension, the truss number of a *node* can be defined as the maximum truss number of its incident edges (Malliaros et al., 2016).

We wrote our own implementations of weighted k-core in R (R Core Team, 2015). For unweighted k-core, we used the `igraph` package (Csardi and Nepusz, 2006), and for k-truss, we used the C++ implementation offered by (Wang and Cheng, 2012).

2.2 Community detection

While the k-core and k-truss decomposition algorithms converge towards the *unique* most cohesive subgraph of a graph, the task of community detection consists in clustering a graph into *multiple* groups within which connections are dense and between which they are sparse (Fortunato, 2010).

Many community detection algorithms have been proposed, of which some of the most popular are listed below. The fundamental *Modularity* function used by the first three algorithms measures the strength of the partition of a graph by comparing the number of within-group edges to the expected such number in a null model (Newman and Girvan, 2004).

The *fast greedy* algorithm (Clauset et al., 2004) merges at each step the pair of nodes that yields the largest gain in modularity until a single community remains. The best partition is the one associated with the greatest modularity value.

The *multi-level* (or *Louvain*) algorithm (Blondel et al., 2008) first aggregates each node with one of its neighbors such that the gain in modularity is maximized. Then, the groupings obtained at the first step are turned into nodes, yielding a new graph. This two-step process iterates until a peak in modularity is attained and no more change occurs.

The *walktrap* algorithm (Pons and Latapy, 2005) uses agglomerative hierarchical clustering with a random walk-based distance between vertices to obtain a set of subdivisions of the graph. The optimal clustering is the level of the hierarchy that maximizes modularity.

Finally, in the *infomap* algorithm (Rosvall and Bergstrom, 2008), optimizes the map equation to find an optimal compression of a description of information flows in the graph. Unlike other aforementioned algorithms, infomap works for directed networks.

We used the R wrappers of the `igraph` C implementations of the algorithms presented above. Note that all `igraph` implementations can (optionally) take edge weights into account. All other parameters remained at their default values.

3 GoWvis

Our system was developed in R Shiny (Chang et al., 2015), and can be broken down into the four modules shown in Figure 2. The steps are sequential except for the last two which are performed in parallel. In what follows, we present the tuning parameters involved at each step and discuss their implementation and individual impact (all other parameters being held constant).

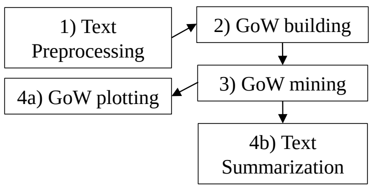

Figure 2: System architecture

3.1 Text preprocessing

The first module cleans the inputted text by (1) removing special characters, punctuation marks except the ones indicative of sentence boundary (used by the second module, see subsection 3.2) and intra-word dashes, (2) removing numbers except dates (like “2016”), and (3) performing tokenization. In addition to R built-in functions, the `stringr` package (Wickham, 2015) is used here. Also, text is split into sentences using the implementation of the Apache OpenNLP Maxent sentence detector offered by the `openNLP` package (Hornik, 2015). The list of sentences is eventually passed to the fourth module (see subsection 3.4). Additionally, the user is provided with the following tuning parameters:

Keep only nouns and adjectives? Boolean, defaults to TRUE. Uses `openNLP`’s implementation of the Apache OpenNLP Maxent POS tagger to perform part-of-speech (POS) tagging. Then, following (Mihalcea and Tarau, 2004), only nouns and adjectives are kept.

Stopwords removal. Boolean, defaults to TRUE. Only actionable if no POS-based screening is performed. Removes common English stopwords from the SMART information retrieval system (accessible via the `tm` R package (Feinerer and Hornik, 2012)).

Stemming. Boolean, defaults to TRUE. Retains only the stem of each term by implementing Porter’s stemmer with the R `SnowballC` package (Bouchet-Valat, 2014). For instance, when stemming is performed, *win* and *winning* are both collapsed to *win*. Stemming thus tends to yield slightly denser and smaller graphs.

The unique words that passed the aforelisted preprocessing steps are then used as the nodes of the graph-of-words.

3.2 Graph-of-Words building

This module constructs the graph-of-words by adding edges between the nodes found at the previous step. It offers the following tuning parameters:

Window size. Integer between 2 and 8, defaults to 3. Specifies the fixed size of the window that should be slided over the document. Values around 3 and 4 have consistently been reported to work well in the literature (e.g., (Mihalcea and Tarau, 2004; Malliaros and Skianis, 2015)). Note that the larger the window, the denser the graph, since more edges are created while the number of nodes remains constant.

Build on processed text? Boolean, defaults to TRUE. Whether the window should be slided over the (1) processed or the (2) unprocessed text. May yield very different results, depending on the preprocessing steps that have been applied. Indeed, two words that are initially very distant in the original, unprocessed text and whose co-occurence would therefore not be captured may end up close to each other in the processed text if many words between them (e.g., stopwords) were removed during preprocessing. Consequently, building the graph from the processed text tends to link more distant words and produce denser graphs than when using the unprocessed text.

Overspan sentences? Boolean, defaults to TRUE. If FALSE, an edge between two co-occurring words is only created (or if the edge already exists, its weight is only incremented) if the two words belong to the same sentence. The punctuation marks ".", ";", "!", "?", and "..." are used here as sentence boundaries.

Color. List, defaults to "heat". A set of five built-in R palettes to color the nodes of the graph, including the color-blind-friendly "gray.colors". Node colors match their core (or truss) number (also indicated in a legend) and go darker as k increases.

3.3 Graph-of-Words mining

This module analyzes the graph-of-words obtained at the previous step using graph degeneracy and community detection. The user can tweak the following parameters to customize the analysis:

Degeneracy. List, defaults to "weighted k-core". Choice of the graph decomposition method, among "k-core", "weighted k-core", and "k-truss". If "weighted k-core" is selected, the edge weights appear as edge labels in the plot.

Directed?. Boolean, defaults to TRUE. Whether edge direction should be taken into account in computing node degree. Only actionable if a degree-based degeneracy algorithm has been selected (i.e., any but "k-truss"). When TRUE, edges in the plot feature arrows indicating their direction.

Mode. List, defaults to "all". Which of the incident edges of a node should be taken into account in computing the node degree, between "all" (all incident edges), "in" (incoming edges only), or "out" (outgoing edges only). Only actionable if edge direction is taken into account, and only impacts the output of the k-core algorithms. Note that the default value "all" gives the same results are when edge direction is ignored but generates a plot with arrow edges.

Community detection? List, defaults to "none". Choice of the graph clustering algorithm, among "fast greedy", "louvain", "walktrap", "infomap", and "none". If not "none", each main community (see *size threshold* parameter below) is separately degenerated. If "walktrap", the user can indicate the length of the random walks between 2 and 8 (defaults to 4). If "infomap", the user can specify whether edge direction should be taken into account. Clustering increases coverage for multitopic documents. It also leads to an increase in processing speed for large graphs, especially if *size threshold* is high.

Weighted? Boolean, defaults to FALSE. Whether edge weights should be used by the community detection algorithm. Only actionable if the community detection parameter is not "none". If TRUE, the edge weights appear as edge labels in the plot.

Size threshold. Numeric (from 0.4 to 1.0, by 0.1), defaults to 0.8. Only actionable if the community detection parameter is not "none". Percentile size threshold used to determine which communities should be considered to be *main* ones. For instance, the default value of 0.8 retains as main communities the ones whose size (i.e., number of nodes) exceeds that of 80% of all detected communities. As will be further illustrated in subsection 3.4, this parameter enables the user to chose whether the summary should cover only the major or also the subtle topics of the document. As *size threshold* diminishes, coverage increases, at the risk of including irrelevant (or noise) topics in the summary.

3.4 Text summarization

The fourth module uses the results from the previous step (graph mining) to (1) extract keyphrases from and (2) select a subset of the original sentences in the document inputted by the user in an unsupervised manner. It is performed in parallel of the graph plotting module (see subsection 3.5).

1. Keyphrase extraction. The terms whose core (or truss) number is exactly equal to $k_{max} - p$ are used as seeds from which keyphrases (n-grams) are reconstructed. p is an integer parameter between 0 and 10 that lets the user navigate the core (or truss) hierarchy up and down. If $p = 0$ (the default), the main core is used. Whenever $k_{max} \leq p$, the user is informed that their selection is empty. In practice, one would want to retain all the words whose core (or truss) number is at least equal to $k_{max} - p$, that is, the members of the

$(k_{max} - p)$-core (or truss), and this is indeed what we do for sentence selection (see "Sentence selection" paragraph below). Here though, we only use a single slice of the hierarchy (called a *lamina*) to make it clear for the user how the process of keyword extraction and keyphrase reconstruction works.

Reconciliation is then performed by pasting together the seeds that are found adjacent in the original, unprocessed text. For example, if "algebra" and "linear" both belong to the selected lamina and "linear algebra" is present in the text, the two seeds are collapsed and added to the set of candidate keyphrase. Duplicates and keyphrases included in higher order keyphrases are removed before printing the results.

When community detection is used, as already explained, each main community is separately degenerated. The entire process of keyterm extraction and keyphrase reconstruction is then ran for each main community, ensuring that keyphrases cover the main topics in the document.

Example. We created a two-topic 925-word document[2] by drawing and intertwining an equal number of sentences from two Wikipedia articles, one about the website *Stack Overflow* (SO) and one about *pizza*. With all default parameters, the keyphrases extracted are all about SO: *stack overflow*, *user*, *answer question*... However, still with all default parameters, by simply enabling community detection (e.g., with "fast greedy"), the two topics are detected (*answer question*, *pizza margherita*, *queen margherita*).

Related work. Similarly, (Bougouin et al., 2013) have used clustering and graph mining for keyphrase extraction, but the other way around. They first group candidate keyphrases into topics via hierarchical clustering (with a word overlap distance), and then apply PageRank on a complete graph with topic nodes and edge weights based on keyphrase offset positions. Closer to our approach is that of (Grineva et al., 2009). Like us, they also observe that terms tend to cluster based on topic and that the largest communities correspond to the main themes in the document. However, they use a complete graph where edges are weighted based on Wikipedia-based semantic relatedness. Additionally, they select *all* the terms in the top-ranked communities whereas we extract only a highly cohesive subgraph from each main group.

[2]https://github.com/Tixierae/examples/blob/master/sopz.txt

2. Sentence selection. Unlike for keyphrase extraction, the entire $k_{max} - p$ core (or truss) is used as seedbed. Representative members are drawn from the list of sentences extracted from the original document (in subsection 3.1) following a three-step process: (1) sentences that do not contain any term belonging to the selected core (or truss) are pruned out, (2) the remaining sentences are ranked in decreasing order according to how many *different* central terms they feature, and finally, (3) sentences are selected one at a time from the top until a certain *summary length* has been reached. If two or more sentences have the same rank, the longest and less redundant one is selected, where length is the number of words in the sentence and redundancy is computed in terms of non-stopword stemmed term overlap with the current summary. Note that we penalize slightly more for redundancy. The *summary length* tuning parameter is a decimal number (between 0.01 and 0.51, by 0.05, defaults to 0.01) indicating the percentage of total candidate sentences (from step 2 above) to include in the summary. Again, if community detection is performed, the process is ran separately for each community, enabling coverage of the main topics in the document. In the previous example, using community detection generates a 11:1 compression ratio summary covering both themes (not shown here due to space limitations).

3.5 Graph plotting

Done in parallel of text summarization. Plots an interactive, dynamic browser-based representation of the graph-of-words using `igraph` and the `visNetwork` R package (Almende B.V. and Thieurmel, 2016).

4 Conclusion and next steps

We have presented `GoWvis`, a freely accessible web application providing an engaging illustration of the GoW concept and how it can be applied to unsupervised extractive single document summarization. Through trial and error, users can navigate the parameter space and develop an intuition as for which parameter values may be optimal for a given task and the particular type of text at hand. Future work should add support for directed degeneracy algorithms (Giatsidis et al., 2011). While showing promise, our summarization approach needs refinement and formal experiments to quantify how it compares to the state-of-the-art. When $p > 0$, taking into account the core (or truss) numbers of terms could yield better sentence ranking.

References

Almende B.V. and Benoit Thieurmel, 2016. *visNetwork: Network Visualization using 'vis.js' Library*. R package version 0.2.1.

Vladimir Batagelj and Matjaž Zaveršnik. 2002. Generalized cores. *arXiv preprint cs/0202039*.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Milan Bouchet-Valat, 2014. *SnowballC: Snowball stemmers based on the C libstemmer UTF-8 library*. R package version 0.5.1.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 543–551.

Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson, 2015. *shiny: Web Application Framework for R*. R package version 0.12.2.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E*, 70(6):066111.

Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, page 16.

Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Ingo Feinerer and Kurt Hornik. 2012. tm: Text mining package. *R package version 0.5-7.1*, 1(8).

Santo Fortunato. 2010. Community detection in graphs. *Physics reports*, 486(3):75–174.

Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. 2011. D-cores: Measuring collaboration of directed graphs based on degeneracy. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 201–210. IEEE.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multi-theme documents. In *Proceedings of the 18th international conference on World wide web*, pages 661–670. ACM.

Kurt Hornik, 2015. *openNLP: Apache OpenNLP Tools Interface*. R package version 0.2-5.

Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. 2010. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888–893.

R Duncan Luce and Albert D Perry. 1949. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116.

Fragkiskos D Malliaros and Konstantinos Skianis. 2015. Graph-based term weighting for text categorization. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1473–1479. ACM.

Fragkiskos D Malliaros, Maria-Evgenia G Rossi, and Michalis Vazirgiannis. 2016. Locating influential nodes in complex networks. *Scientific reports*, 6:19307.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113.

Pascal Pons and Matthieu Latapy. 2005. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer.

R Core Team, 2015. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: new approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 59–68. ACM.

François Rousseau and Michalis Vazirgiannis. 2015. Main core retention on graph-of-words for single-document keyword extraction. In *Advances in Information Retrieval*, pages 382–393. Springer.

François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In *ACL*, volume 15, page 107.

Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks*, 5(3):269–287.

Jia Wang and James Cheng. 2012. Truss decomposition in massive networks. *Proceedings of the VLDB Endowment*, 5(9):812–823.

Hadley Wickham, 2015. *stringr: Simple, Consistent Wrappers for Common String Operations*. R package version 1.0.0.