UCLouvain

epl

École polytechnique de Louvain

# Analysis and comparison of literature dedicated visual mapping software

Author: **Brieuc Pierre**
Supervisor: **Yves Deville**
Readers: **Benoît Duhoux, Christine Jacqmot, Charles Pecheur**
Academic year 2022–2023
Master [120] in Computer Science and Engineering

# Acknowledgements

I would like to convey my sincere gratitude to Pr. Deville and Dr. Jacqmot for their constant assistance and guidance throughout this journey. Many thanks for the time they dedicated to our monthly meetings and for their constructive feedback, all of which have made this thesis possible.

I am grateful to the other members of the jury, Pr. Pecheur and Dr. Duhoux, for serving on my thesis committee. I am also grateful to Ms Jamar for her help in solving technical problems

I would like to acknowledge and thank my parents for their constant support and encouragement. Special thanks to my father for proofreading this thesis[1].

Lastly, I would be remiss in not mentioning Mira for her emotional support and motivation along the way.

[1] DeepL Write was also used to improve the writing style of this thesis. DeepL is a deep learning company that develops AI systems for languages.

# Abstract

With millions of literature documents available online, exploring them wisely becomes an almost impossible challenge. Literature dedicated visual mapping software rose from this challenge. This thesis aims to investigate some of the most advanced software in this domain: Open Knowledge Maps, Open Research Knowledge Graph, InfraNodus, Connected Papers, Litmaps, Inciteful and Research Rabbit. A literature review presents the workflow for each of these software. A case study provides a common background for analysing and comparing the aforementioned software that produces knowledge maps whose map components are literature documents. The results suggest that Research Rabbit uses a more relaxed search when selecting documents for its map, and that the different nature of Open Knowledge Maps also provides a more varied selection of documents than the other software. This study shows that users should be cautious about the data found in the maps and recommends using more than one software to avoid any bias that a single software might induce.

# Contents

# List of Tables

# List of Figures

# Nomenclature

| | |
|---|---|
| CP | Connected Papers |
| ICF | Inciteful |
| LM | Litmaps |
| OKM | Open Knowledge Maps |
| ORKG | Open Research Knowledge Graph |
| RR | Research Rabbit |
| std | Standard deviation |
| TSP | Travelling Salesman Problem |

# 1 Introduction

Due to the exponential growth of scientific knowledge and the systematic digitisation of scientific documents, millions of documents are now available online and many more are added every day. With so many resources available, it has become increasingly difficult to obtain a good overview of the relevant information on a particular topic using tools such as Google Scholar, Semantic Scholar, etc. The problem with these conventional tools lies in the format of their output. When queried, they respond with pages containing long lists of documents sorted according to criteria to be selected by the user. Furthermore there is no information about the possible correlations between these documents. It is impossible to find out if a document is referenced in another document without going through its bibliography. For the same reason, it is impossible to realise how much two documents correlate without reading their contents. Meticulous research is therefore becoming increasingly time-consuming and difficult. The need for an efficient tool to help the user gain a useful overview of his or her subject of interest was therefore growing by the day.

Two concepts have been used to address this need: knowledge graphs and knowledge maps. Both concepts aim to use the data or metadata of the above-mentioned documents to provide the user with a visual representation of the accessible documents. The first solution, knowledge graphs, looks at the content present in each of the accessible documents in order to create a graph that contains all the knowledge offered by these documents. The nodes of such graphs are often defined as words that are linked together. The resulting representation is a network of interlinked words, representing the topics present in the accessed content and how they relate to each other. An example can be seen on FIGURE 1.

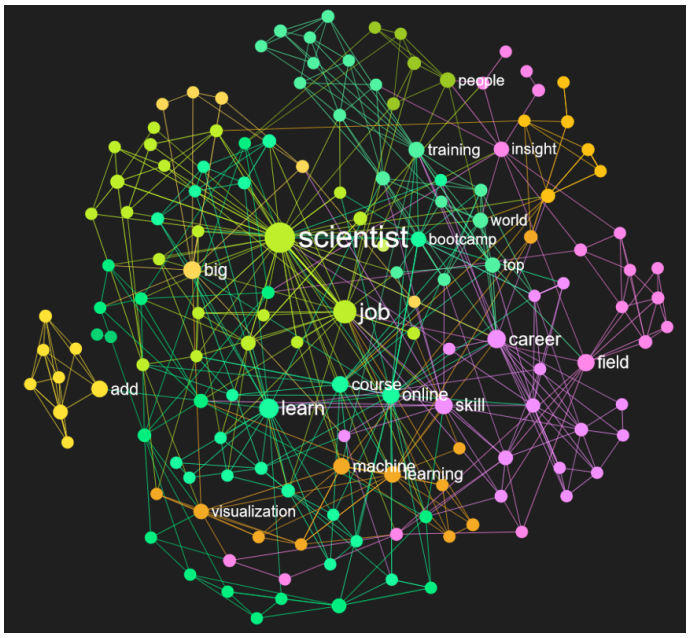



Figure 1: Knowledge graph example taken from the tool InfraNodus

The latter solution, knowledge maps, on the other hand, doesn't necessarily need the full content of the documents and can only work with their metadata. With the information it has access to, it aims to create a map that shows the relationships between the documents. The placement of documents within the maps would then define clusters of related documents. An example can be found in the underlying FIGURE 2.

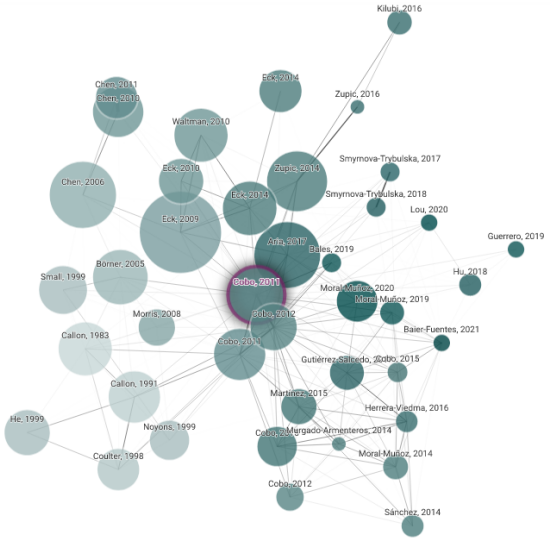



Figure 2: Knowledge map example taken from the tool Connected Papers

Various software have been developed for both concepts, although knowledge maps have gained more popularity in the context of scientific literature research. This thesis will mainly focus on software based on this concept of knowledge mapping and will aim at two objectives. The first objective is to present an overview of the current progress in the field of visual mapping by achieving a survey. This will also allow to select software that put forward different approaches. This objective will be pursued in the literature review laid down at Chapter 2 where each selected software will be explained. The second objective is to select software with comparable results in order to be able to evaluate them. Since all of these software programs serve the same purpose, it is indeed difficult to decide which one to use. The comparison of the selected software is carried out in the Analysis and comparison Chapter. Both objectives are then discussed in Chapter 4 to help users decide which software to choose to get an overview of their topic of interest.

# 2 Literature review

Many different visual mapping software are available online. This Chapter selects the most commonly used ones that attempt to add value to the data they are able to collect. Software that only generate citation trees, a network of documents where each edge between two nodes (documents) represents a "cited by" or "referenced by" relationship, don't need much to talk about. They are just visual representations of raw metadata, and their consistency depends only on their data providers. The remaining software will be explained in detail in this Chapter.

## 2.1 Software categorisation

As mentioned in the introduction, visual mapping software aim to visually represent a collection of information. The output of all considered software will therefore serve the same main purpose and will show some similarities. However, the processes used to produce these outputs may be more different. The core of these processes lies in the data they use. Based on the use of data, two categories of software can be identified. The first category uses text-based metadata such as abstracts, authors, titles, etc. The second category uses both the citations and the references of the literature. Both categories of software, respectively classified as "text-based approach" and "citation approach", have a common structure, which is outlined in the Process Structure Section, while the differences can be observed in their respective Sections.

Although the correlation between the two categories could be of interest, the nature of the software doesn't allow to rigorously investigate this avenue. This is due to the fact that most of the software are not open source and don't provide enough information about the provenance of the data or how they are used. Hence the aim of this work is to assess the quality of the software's output rather than the relevance of the data. It is important to note that there are other approaches, such as co-authorship, to link documents with common authors, or co-readership, to link documents that are read by the same users. These approaches are not widespread practices, so they are left out.

## 2.2 Process structure

Although the steps taken by the processes of the categories identified above differ, their overall structure remains the same. The first step in querying any of these software is for the user to provide some initial content. Depending on the software, this may be either a document or a text query with optional filter parameters. In response to the user's query, the software must have access to some data from other documents. Most software would typically be connected to one or more data providers that have indexed large amounts of document metadata. The data providers can only store and share the metadata of the documents they have indexed, because the cumulative storage size needed to store millions of documents and their content is not yet possible with current technologies. It would require too much capacity and would be too expensive. The other software would rely on its users to provide it with the necessary data. With the data collected, the software is able to perform

calculations on it to identify the most relevant documents. Performing calculations on the data of millions of documents would take a very long time, which doesn't suit the user. To limit this, the software will always limit the number of documents it will analyse. An amount that depends on the type of data the software exploits and the complexity of its algorithm. At the end of the processing time, the most relevant documents are presented visually to the user. The following FIGURE is an illustration of the outlined workflow.

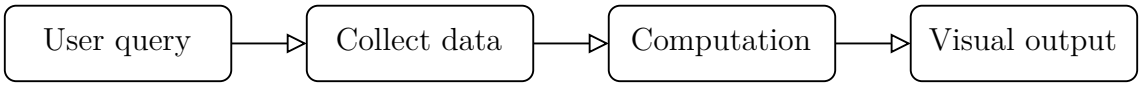



Figure 3: Process structure workflow

## 2.3 The text-based approach

As previously explained, there are millions of documents available online, making it impossible to search the content of all or even a subset of them in a reasonable amount of time. For this reason, text-based software rely only on their metadata. This doesn't take into account the fact that a significant proportion of these documents are not open access. But even then, the metadata may still contain a lot of exploitable data, such as: title, abstract, author, year of publication, keywords, publisher's institution and country, etc.

Three software based on the text-based approach have been selected in the scope of this work: Open Knowledge Maps, Open Research Knowledge Graph and InfraNodus. They will be described in the following subsubscetions.

### 2.3.1 Open Knowledge Maps (OKM)

OKM is a visual interface that focuses on classifying and organising scientific literature into clusters (subdomains) in order to provide the user with a clear overview of a topic (domain). The purpose of this visual interface is to enable the exploration of existing knowledge and the discovery of new knowledge [2].

To this end, they have integrated two data providers, both of which have indexed the metadata of millions of documents: PubMed and BASE. The focus will be on the latter, as the former focuses only on biomedical and life sciences literature. BASE provides over 300 million documents from over 10,000 content providers. Approximately 60% of these documents are open access.

The process of creating a map is illustrated in the underlying FIGURE:

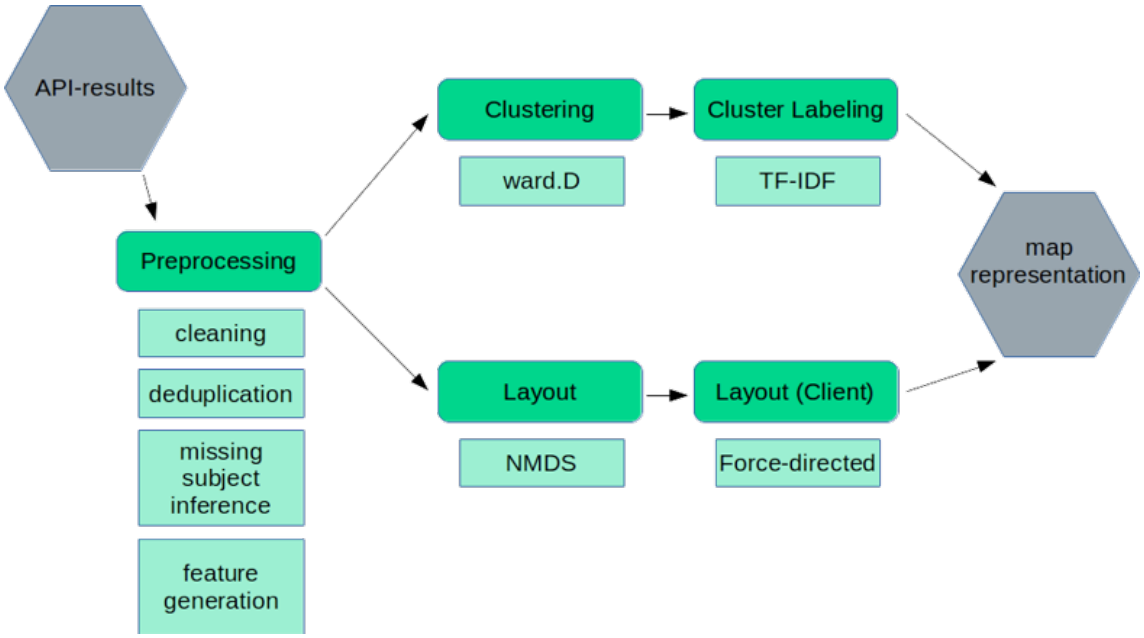



Figure 4: OKM workflow

Each query is fed to the selected data provider, in this case BASE, which responds with the metadata of the 100 most relevant documents using its own ranking algorithm [3]. OKM will filter these documents further by removing any document whose abstract exceeds 300 characters.

The remaining data are pre-processed by filtering for stop words ( words that hold no meaning such as: it, and, a, the, etc.), removing punctuation and stemming [4]. At this point, a bag-of-words for each document is transformed into a frequency word (stem) vector. The pairwise cosine similarity of the documents is then computed by comparing their respective vectors [5]. The values obtained can be represented in a square similarity matrix where the indices of the rows and columns uniquely represent each document.

At this point, the algorithm can simultaneously identify clusters of documents and place them on the map.

The clustering of documents is done by Ward's method of minimum variance, a bottom-up hierarchical clustering [6]. The aim is to minimise the total variance within the cluster. Each document starts as an independent cluster, and after each step, the pair of clusters that gives the best objective value is merged. The number of clusters is determined by the elbow method, which chooses a number of clusters k that explains at least 80% of the model variance and where the increment for k+1 is less than 1%. A maximum of 15 clusters is set, regardless of k. This method tends to produce clusters of approximately the same size.

Placing the documents on the map is perfomed by non-metric multidimensional scaling (NMDS). The set of procedures known as multidimensional scaling (MDS)

methods aims to construct a configuration of n points, usually in a Euclidean space, based on information about the pairwise distances between a set of n different documents [7]. NMDS is a subcategory of MDS that deals with non-numerical distances between documents.

Once the clusters have been identified, they are placed on the map using the average of the coordinates of their documents based on the results of the NMDS [8]. The name of each cluster is found by sending all its documents to the Zemanta [9] and OpenCalais [10] API. These return the main concepts (words) of the documents' metadata. The name of each cluster is the combination of the top three concepts that receive the highest weights from the TF-iDF approach within the cluster itself [11].

For better visualisation, a force-directed layout optimisation is performed to reduce overlap and increase the distinctiveness between clusters. In this iterative method, each document is considered to be connected to the other documents within its cluster by physical strings. Step by step, the model determines the forces acting on the documents and moves them to come as close as possible to equilibrium. This means that the positioning of documents within a cluster no longer represents their similarity.

The map's interactions focus on Shneiderman's principle of "overview first, zoom and filter, then details-on-demand", which provides many visual design guidelines and a framework for designing information visualisation applications. The map is also given a unique identifier and is stored in a SQLite database for consistency of presentation.

### 2.3.2 Open Research Knowledge Graph (ORKG)

ORKG is a software that provides knowledge-based graphs. Unlike the other software presented in this Chapter, it relies on a mixture of manual and automated techniques [12]. Information scientists have implemented a reusable infrastructure to make it easier for domain experts and researchers to populate ORKG. They believe that collaboration between information scientists and domain experts with the help of automated tools promotes precision and sharpness against ambiguity.

The following user story explains how ORKG is populated [13]. Anyone who wants to contribute can add a new document to ORKG. Tools such as [14] attempt to extract the metadata, which needs to be validated by domain experts. After adding a document, ORKG asks the contributors to select the field of the document and fill in their contribution. Other (semi-)automated tools such as [15] and [16] help to extract the data. A contribution is a minimal domain-specific structure that identifies the important information of similar methods for later comparison. An example of a comparison can be seen on Figure 5. Each column represents a contribution from a document and each row represents a common field of those contributions.

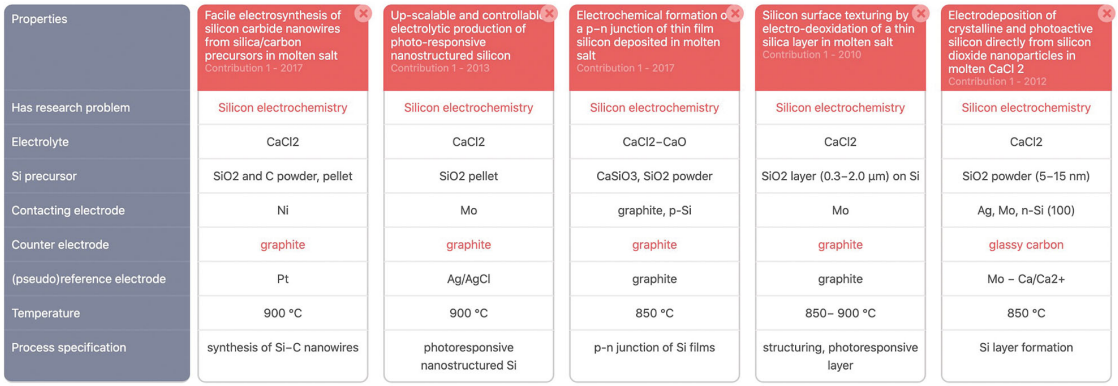



Figure 5: ORKG comparison example

Since the fields of a contribution can still vary between different fields, it is necessary to use an algorithm [17] to find similar fields and thus closely related contributions. Contributions are selected if they share enough similar fields (row header in FIGURE 5). The similarity is calculated using the cosine similarity between the word vectors of the fields [18]. The resulting value must be above the given threshold to be considered similar. This makes it possible to compare different documents, since their contributions have been extracted by a contributor.

The infrastructure chosen for the storage and use of the collected data makes it difficult to query ORKG as other research tools such as Google Scholar, Semantic Scholar, etc. This requires that the contributions adopt a comparable format. For this purpose, each contribution is converted into a document that is compared with the query using the TF-iDF approach [11].

### 2.3.3 InfraNodus

InfraNodus is a visual mapping software that has a different focus than the other software. Instead of mapping documents, it maps concepts (words). This software is interesting because it can take literature content as input to produce its network. The user is asked to add texts manually and is given feedback to help him get a complete overview of the domain of his choice. The output of InfraNodus is a network where the nodes are words and the edges represent their co-occurrences. In order to be able to generate the feedback, the software first performs an analysis of the network, which will be discussed in this Subsection. An example of a map can be seen in the introduction in the FIGURE 1. Most of the information here comes from the [19] and InfraNodus documentation [20].

The first step before creating the network is to transform the input text into an exploitable structure. As with the OKM software (see Subsection 2.3.1), the stop words and punctuation are removed. Unlike the OKM software, InfraNodus uses lemmatisation instead of stemming. The use of this algorithm increases the accuracy at a worse time complexity, as [4] explains. At this point there is a sequence of lemmas, each sequence representing a paragraph. These lemmas will be the nodes of the network.

The weighted edges represent the co-occurrence frequencies between pairwise nodes. The weights are determined using 4-grams, i.e. for each sequence of lemmas, a context window containing four consecutive lemmas is progressively moved. For each iteration, the first lemma is considered to co-occur with the other lemma within the current context window. The following example shows in orange underlined text two possible iterations for a sequence of 5 lemmas:

iteration 1 : lemma1, lemma2, lemma3, lemma4, lemma5
iteration 2 : lemma1, lemma2, lemma3, lemma4, lemma5

The weights are adjusted to express the closeness of the co-occurrences (i.e. lemma1 is closer to lemma2 than lemma3). Once the network is created, it is possible to identify its centre and its clusters. The central nodes can be found using the betweenness [21]. The more a node is within the shortest path between two random nodes, the closer it should be to the centre. A community detection algorithm [22], [23] is used to select the clusters by identifying groups of highly connected nodes. The final step is to construct the network in a visually understandable way using the algorithm from [24]. This algorithm pushes the most connected nodes (i.e. hubs) apart, while gathering the other nodes towards their connected hubs. Nodes from the same aforementioned clusters will often be visually close, as the [23] and [24] algorithms are correlated. More details, beyond the scope of this thesis, can be found in these publications [25].

So far, the content of the resulting network only allows to find the main concepts (keywords) of the documents. However, its structure can be used to identify the weaknesses of the network. To do this, InfraNodus has adopted a so-called ecological thinking process. The idea behind this process is to consider the input texts as a dynamic ecosystem of interdependent parts, with the aim of bringing them into play by inducing variability in two spectra: scale and intention. The former balances diversity and coherence by zooming in and out of the network. The latter ensures a constant flow of new ideas while correctly connecting them to the current network. Together they define a looping process that never gets stuck (see FIGURE 6).

The following figure defines a Cartesian plane representing the possible states of the ecological thinking process. The horizontal axis describes the degree to which the concepts are connected within the network. The more connected they are, the further the state is from the origin of the plane. The vertical axis describes the focus of the software. The more concepts are in focus, the higher the state. If the current network is densely connected, while InfraNodus focuses on a single concept, then the state of the process is in the lower right part of the plane and the next step would be "diversification" as shown in the plane. This means that InfraNodus will start to focus on more concepts (zoom out) and explore new concepts that won't be connected to the network, thus decreasing the connection density.

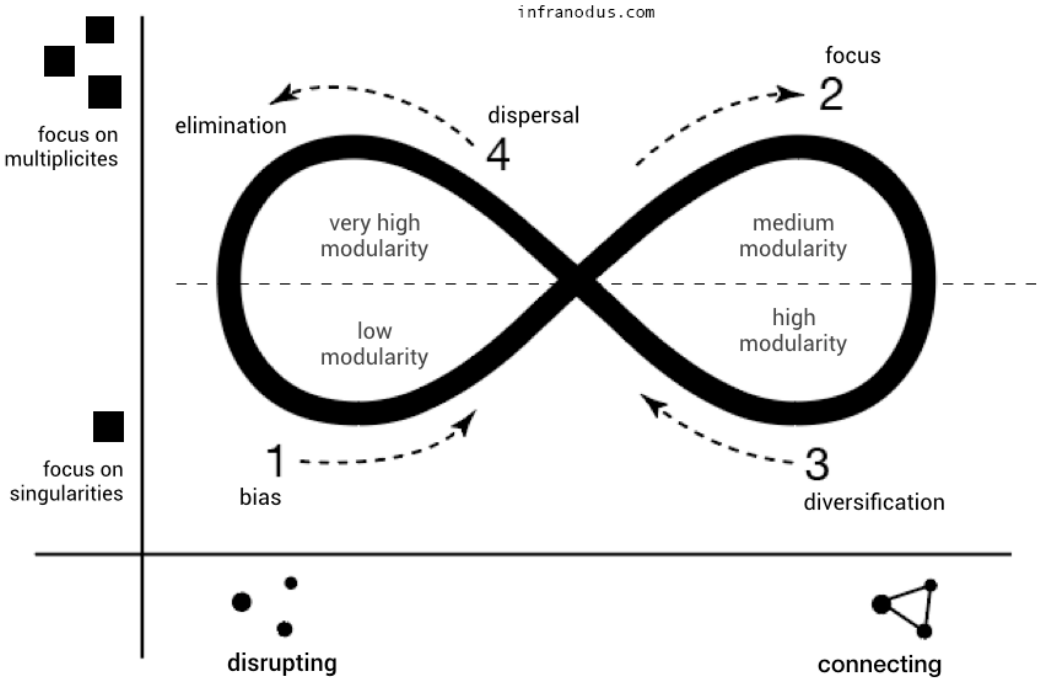



Figure 6: InfraNodus ecological thinking process

Starting with a very simple network by only adding the abstract of a single document and the network state in the middle left of the figure above. The first step would be to zoom in on one of the least connected new concepts in the abstract. The algorithm will ask the user to provide more data (e.g. another abstract) on this concept to gradually connect it to the rest of the network while increasing the scale (the concept plays a role in both intra- and inter-cluster connections). After a few iterations, the concept of focus will be saturated, at which point InfraNodus will suggest to the user to develop on the specificity of older concepts. A few iterations will be made, starting on a small scale to fill the gaps and slowly increasing the scale to explore new ideas. A new iteration of the loop can then be undertaken. A more detailed workflow can be found in FIGURE 14 at Annex.

The primary function of this ecological thinking process is to induce an optimal level of coherence, but it is up to the user to use it in this way. It is also possible to enforce a certain bias or diversity. A possible use case is to add one or more abstracts of documents from a domain of interest. Depending on the state of the network, InfraNodus will suggest to explore or focus on certain concepts that can be searched with a search tool such as Semantic Scholar or Google Scholar. The resulting information should gradually accumulate into a set of documents that provide a complete and broad overview of the domain.

## 2.4 The citation approach

Comparing documents based on their textual content is quite tedious, as you may have guessed from reading the previous Section 2.3. Another way is to look at citations. There are several metrics to measure the proximity of documents. The two most commonly used are: co-citation [26] and bibliographic coupling [27]. The former links documents that are cited by the same set of documents. The latter links documents that reference the same set of cited documents. Both approaches produce relevant results, as [28] shows. The following FIGURE shows these relationships. The arrows represent the "cites" relationships.

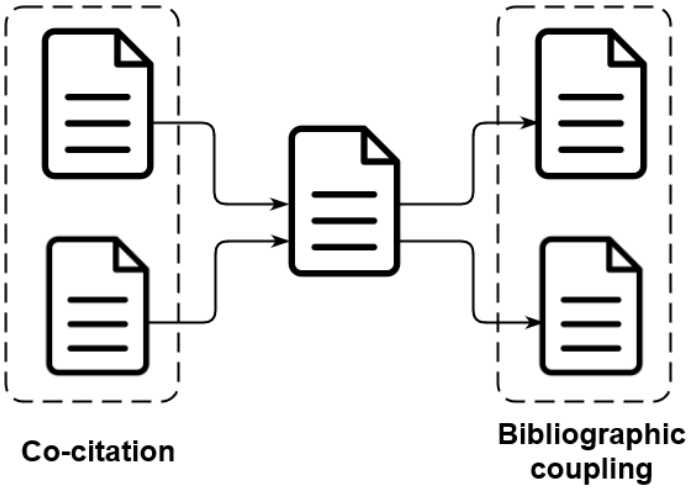



Figure 7: Co-citation and bibliographic coupling diagrams

The queries for these software are composed of one or more documents, also called seed documents, around which the software build a map. In this approach, a link between 2 documents always represents a citation relationship. A citation network is always a directed acyclic graph because of the temporal constraint on citations. It is impossible for a document to cite another document that was written later, which means that the edges of the maps are directed and never bidirectional. This characteristic means there can't exist any loop inside citation networks which allow software to obtain better time complexities than regular network analysis software.

Four software based on the citation approach have been selected in the scope of this work: Connected Papers, Litmaps, Inciteful and Research Rabbit. They will be described in the following Subsections.

### 2.4.1 Connected Papers (CP)

CP is a unique visual tool to help researchers and applied scientists find and explore documents relevant to their field of work [29]. It creates a map based on a single seed document, to which several other seeds can be added to specialise the initial query. The documents are provided by Semantic Scholar and those with the closest similarities to the seed documents are used to create the map. The similarity metric

is based on the concepts of co-citation and bibliographic coupling (i.e. the more the citations and references of two documents overlap, the more similar they are). Like with the OKM software (see Subsection 2.3.1), the map uses a forced layout.

### 2.4.2 Litmaps (LM)

LM is a browser-based research platform designed for clarity, comprehensiveness and collaboration [30]. It can directly create maps from multiple seed documents and uses Crossref, Semantic Scholar and OpenAlex as sources. They use metrics such as bibliographic coupling, co-citations. LM uses a 2 degrees search depth. The first degree identifies the set of documents directly cited or referenced by the seeds. A scoring system is then used to filter out the least common documents. Each additional degree of depth is the product of applying the same logic, but to the set of documents from the previous iteration. This results in a massive edge set that provides a much broader search coverage, identifying articles that are shared in more subtle ways than direct citations. It is possible to organise the map along different axis metrics: number of citations, number of references, publication date, aesthetic layout (to provide the least obscured layout of articles to make it easy to assess how articles are related), number of citations weighted by publication date, or number of citations within the visualisation.

### 2.4.3 Inciteful (ICF)

ICF is a visual tool for exploring documents relevant to a researcher's field of work. It uses OpenAlex, Crossref, Semantic Scholar and OpenCitations as data providers. Like the LM Subsection 2.4.2, they perform a 2-degree depth search from the seed documents. The second degree is only performed if less than 150 thousand documents are identified after the first iteration. The resulting set of documents is filtered by importance or similarity using the PageRank [31] and Link Prediction [32] algorithms. PageRank is not only good at finding the most cited documents (often biased towards older documents), but also the most influential. Link Prediction uses the concepts of bibliographic coupling and co-citations (both often biased towards more recent documents). Note that both of these metrics can have problems if the number of citations varies widely and is biased towards the publication years of the documents. The Adamic/Adar algorithm [33] is chosen for the bibliographic coupling metric, while the Salton Index algorithm (cosine similarity) is chosen for the co-citation metric. Both try to compensate for the biases and asymmetries that can be found in citation networks.

### 2.4.4 Research Rabbit (RR)

ResearchRabbit is a research platform that allows the user to discover and visualise relevant literature [34]. It can directly generate maps from multiple seed documents and consolidates multiple databases (unnamed). The metrics used to generate the maps are based on citation network analysis, but no additional data are provided.

# 3 Analysis and comparison

As mentioned in the general introduction, the second objective of this thesis is to compare software of the same nature. Indeed it is not possible to compare each of the software selected from the literature review as they do not all output the same type of knowledge. This thesis will compare the software that provides knowledge maps, as knowledge graphs do not really solve the problem identified in the introduction, which is to help users get a good overview of the literature on a topic of interest. In fact, knowledge map tools focus on the relevant documents, whereas knowledge graph tools focus on their content, resulting in a visual structure that mixes the content of the documents. The relevant software are therefore Open Knowledge Maps (OKM), Connected Papers (CP), Research Rabbit (RR), Litmaps (LM) and Inciteful (ICF). All of these are free to use and provide a visual map of documents from the literature when queried.

In order to provide an objective comparison of the above software, it is necessary to define the methodology used to carry it out. This will be done in the next Section 3.1. The rest of the structure of this Chapter represents different aspects or levels of comparison.

The simplest aspect is the statistical analysis of individual numerical fields of the metadata present in the csv file (i.e. the 3.2 and 3.3 Sections), filtered by software. For the text fields of the metadata, such as the titles or the abstracts, additional steps are required in order to perform any statistical analysis. The Section 3.4 looks at the contents of the abstract text field using an algorithm described in it. Going a step further, comparisons can also be made by looking at multiple metadata fields at the same time. The Sections 3.5 and 3.6 do this. Unlike the 3.4 Section, they deal with text through a black and white comparison, i.e. either two texts are identical or they aren't.

To assess the overall quality of a map, the Sections 3.7 and 3.8 go one step further by focusing on the visual aspects of the map.

The 3.9 Section also plays an important role, as it will have an impact on the user experience and ease of use of the software. The features have been broken down into simple user stories.

## 3.1 Methodology

The aim is to objectively compare different visual mapping software. This can be done by assessing the quality of the data contained in the maps produced and by considering the usability of each of these maps. This Section sets out the procedure for comparing software.

In order to obtain the best conditions for comparison, it is important to homogenise the queries of the selected software. In fact, the query format of OKM is different from the others. The former takes a text query, as a search engine would do (e.g.

Semantic scholar, Google scholar, etc.). The latter take one or more documents. The effect of this difference can be divided into two correlated characteristics: the quality of the query and of the corresponding map. In the case of OKM, if the query is too specific, the software won't be able to identify enough documents to create a map. Conversely, if the query is too vague, the map is likely to include many topics that are of no interest. To minimise this effect, OKM's query must find the best trade-off between the two. For the other software, the characteristics mentioned above are mainly influenced by the number of citations. If the input documents of the literature have low counts, the generated maps won't be able to take advantage of the co-citations nor of the bibliographic coupling.

Based on the these differences, two requirements are defined to provide the best environment for the comparison. For OKM, the requirement is that all the input documents in the other software must be present in OKM's map. For the other software, each input document must be cited by at least 100 other documents. Therefore, a single document of literature is used to generate the maps. The selected document is [1] with 322 citations and 65 references (per Semantic Scholar). The matching query input in OKM is "travelling salesman problem drone".

All generated maps are made up of documents whose connections are shown visually. In order to assess the difference in quality between each map, it is necessary to harvest the metadata of their documents. These metadata are stored in a common csv file, the structure of which is shown in the example below.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Title | Description | Citations | Year | Tool |
| 104 | On the cor | A fleet of uni | 41 | 2013 | RR |
| 105 | Feature su | Practical patt | 1226 | 1998 | RR |
| 106 | On the Mir | TL;DR: A new | 318 | 2015 | CP |
| 107 | Optimizati | The fast and | 453 | 2016 | CP |

Figure 8: Extract from the csv file

If an item of literature is present in several maps, it will appear in the csv file as many times as its number of occurrences in all maps. It is a necessary redundancy for the Data Correctness Section. Unless explicitly stated, the data presented in this thesis has been taken directly from the selected software. The maps generated for this Chapter can be found in the annexed maps.

## 3.2 Entries' year of publication

The year of publication plays an important role when exploring a domain. The most recent documents may reflect the latest discoveries, while the older ones may be the basis for them.

| | Open Knowledge Maps | Connected Papers | Research Rabbit | Litmaps | Inciteful |
|---|---|---|---|---|---|
| \|documents\| | 53 | 41 | 50 | 20 | 31 |
| median(year) | 2021 | 2019 | 2016 | 2020 | 2018 |
| std(year) | 1.81 | 1.59 | 4.59 | 2.29 | 2.20 |

Table 1: Statistics on the entries' year of publication by software

RR has a lower median but a higher standard deviation (std). The range of its distribution goes from 1998 to 2017. Half of the Litmaps documents were published in 2022. The other software have a similar distribution. RR and ICF are the only software with documents that were published before 2015 (21 documents for RR and 4 documents for ICF). The detailed distributions are shown bellow.

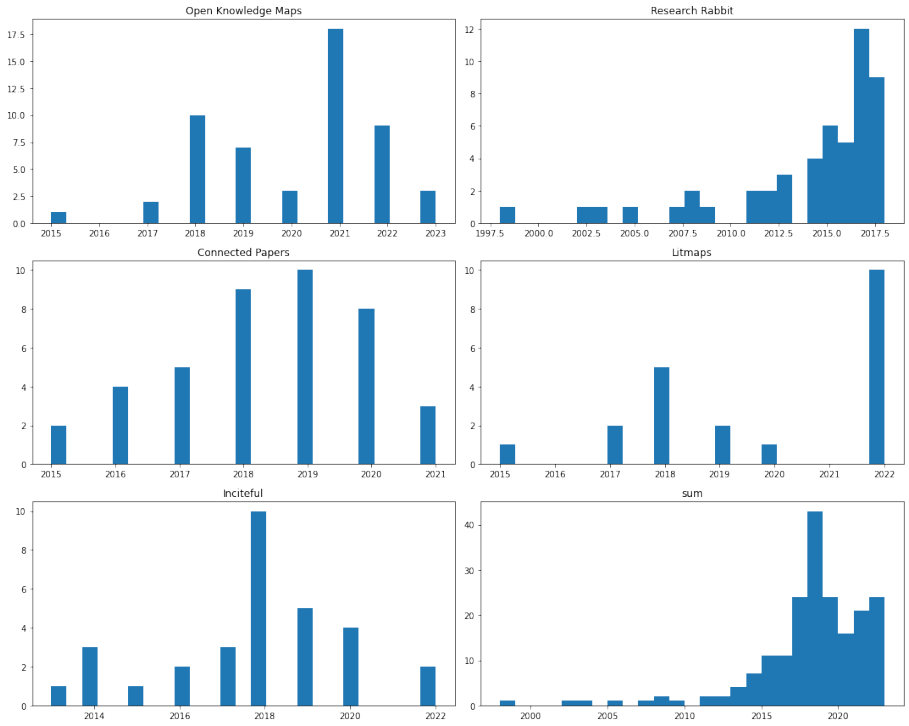



Figure 9: Publication year distribution by software and overall

## 3.3 Entries' citation counts

For this Chapter OKM uses as source Base search, which doesn't provide the citation counts of the entries. For the sake of comparison, Semantic Scholar is used to collect the missing information. The underlying table shows several metrics about citation counts. Each value is calculated separately, without looking at the other columns, except for the percentages. It provides a simple means of comparison by dividing the value directly above by the sum of the same row.

| | Open Knowledge Maps | Connected Papers | Research Rabbit | Litmaps | Inciteful |
|---|---|---|---|---|---|
| \|documents\| | 53 | 41 | 50 | 20 | 31 |
| sum(citations) | 2228 | 6512 | 12013 | 2220 | 3193 |
| median(citations) | 9 | 103 | 110 | 85 | 72 |
| std(citation) | 89.87 | 154.30 | 382.52 | 141.59 | 124.84 |
| mean(citations) | 42.09 | 158.83 | 240.26 | 111.00 | 103.00 |
| % | 6% | 24% | 37% | 17% | 16% |
| documents in top tier (citations > 115) | 5 | 18 | 24 | 7 | 11 |
| \|documents in top tier\| /\|documents\| | 0.09 | 0.44 | 0.48 | 0.35 | 0.35 |
| % | 5% | 26% | 28% | 20% | 21% |

Table 2: Statistics on the entries' citation counts by software

The rows with the % symbol in the $5^{th}$ and $7^{th}$ criteria are used to explain the percentage of each cell value compared to the sum of all values in that criterion.

$$\texttt{eg:} \quad \frac{42.09}{42.09 + 158.83 + 240.26 + 111.00 + 103.00} = 6\%$$

RR has the highest standard deviation because its map contains the most cited documents (see FIGURE 10). OKM stands out with only 5% in the normalised metrics because the majority of its entries are cited less than 10 times. Once the scales are adjusted, LM and ICF show very similar results with around 16%. RR has the best performance and CP is in between. OKM is the only software where less than 20% of its map documents are in the top tier of most cited documents (all maps documents considered). This shows that OKM does not take citation counts into account (and does not have access to them). The distribution of citation counts may help to understand the reason for these results. FIGURE 10 shows the distribution for each of the software with the additional overall distribution (bottom right graph).

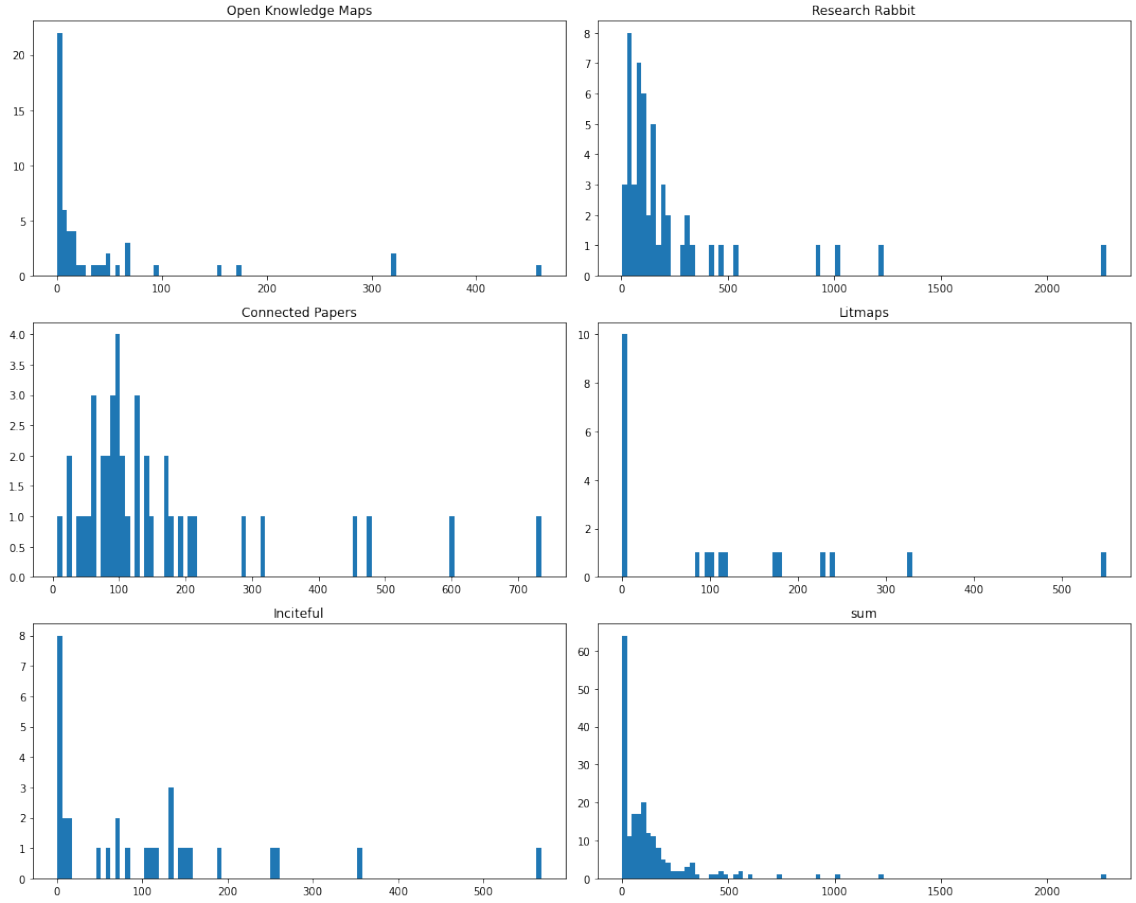



Figure 10: Citation count distribution by software and overall

OKM, LM and ICF all have many documents that aren't cited, which lowers their statistics.

## 3.4 Entries' abstracts' similarity

This Section uses a small annexed Python program to calculate a value representing the similarity between two abstracts. This algorithm uses lemmatisation and TF-iDF to transform an abstract into an array where each value represents its frequency within that abstract, normalised by its occurrences across all abstracts in the csv file. The closeness can then be found by finding the cosine similarity between two arrays. The table below shows how close entries tend to be to each other within a single map.

| | Open Knowledge Maps | Connected Papers | Research Rabbit | Litmaps | Inciteful |
|---|---|---|---|---|---|
| mean(closeness) | 13.34% | 8.96% | 7.21% | 9.51% | 12.01% |
| median(closeness) | 12.16% | 7.95% | 5.84% | 6.64% | 11.51% |
| std(closeness) | 7.74% | 6.70% | 7.20% | 7.94% | 8.02% |

Table 3: Statistics on the entries' abstracts' similitude by software

OKM and ICF have a better overall proximity. Again, the statistics may not be enough to understand how each software performs. FIGURE 11 shows how many links represent a closeness between two entries against a linear threshold.

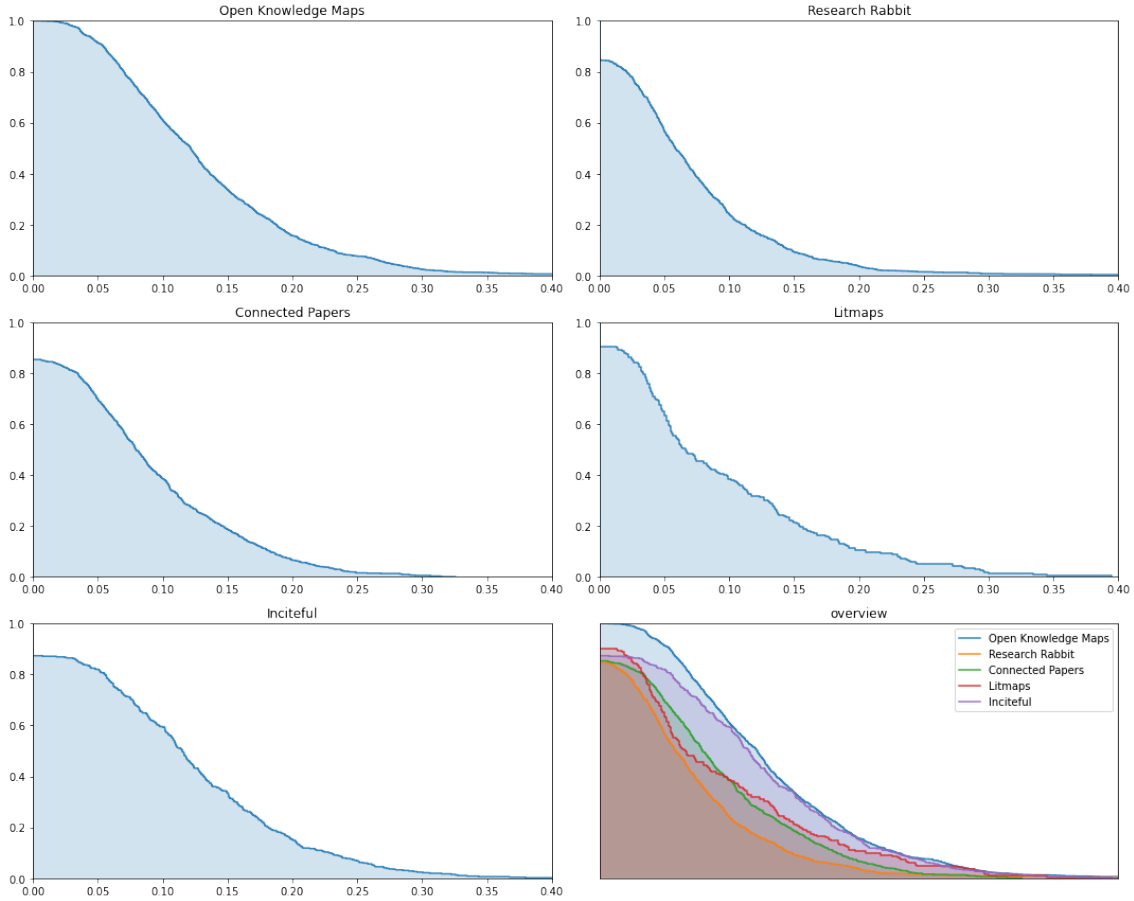



Figure 11: Number of entries normalised against similitude within each map by software

The x and y axes describe the similarity threshold and the proportion of links between two documents that are above this threshold, respectively. OKM is the only software where every document has a similarity above 0 with the rest of the documents. This is probably due to the fact that the other software focus on the citation approach rather than the text-based approach. As can be seen in the lower right graph, OKM has the highest overall similarities. ICF follows the same trends as OKM, except for a few documents with 0 similarities. LM also has highly similar documents like OKM and ICF.

Another interesting approach is to look at the overall proximity of each map's entries with [1]. FIGURE 12 plots the number of entries within a map against its closeness threshold relative to [1].

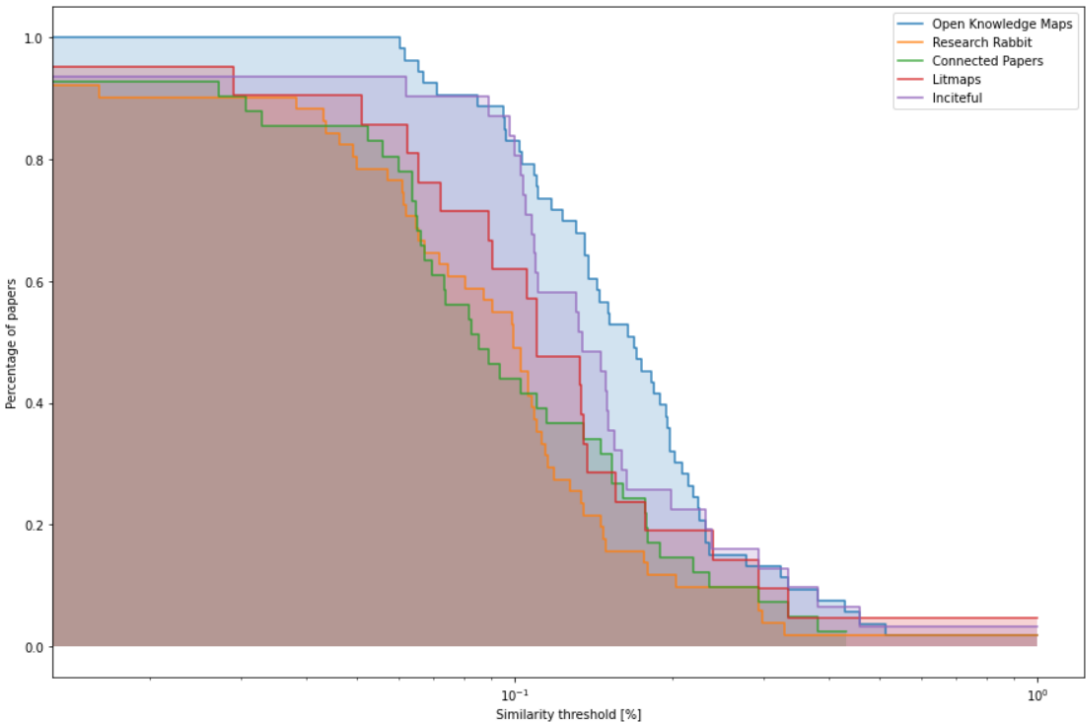



Figure 12: Number of entries normalised against similitude threshold with [1]

With a threshold above 20%, only a tiny fraction of entries remain for each map. Because the TF-iDF algorithm normalises the weights of terms by their occurrences in all entries, a small cosine similarity doesn't mean that two abstracts are unrelated. As the entire set of entries is linked to [1], this approach only allows the values to be compared with each other. OKM has the best performance. It focuses on metadata proximity (without citation metrics), while the others prefer to exploit citation relationships.

## 3.5 Data Correctness

Each software uses its own sources to search across documents. One concern is that in-homogeneous sources can lead to inconsistent metadata for the same documents. The following table shows how many entries in the csv file have at least one identical metadata attribute for the same document:

| | Title | Abstract | Citations | Year |
|---|---|---|---|---|
| \|csv entries\| | 79 | 30 | 4 | 74 |

Table 4: Number of entries with at least one identical metadata attribute for the same document

As can be seen, 79 entries share a title with at least one other entry, while this is not the case for the other attributes. As will be discussed, these differences are often related to the difficulty of extracting the attributes or the age of the data. Note that there are a total of 82 of documents that occur multiple times in the csv file. In total the file contains 143 different documents. This means that about 27% of the entries are not unique to a document.

Some inconsistencies found in the maps, but not directly covered by the csv file, include ICF incorrectly labelling documents as "not open source" and CP truncated abstracts labelled as "too long, didn't read".

## 3.6 Documents' overlap

When comparing entries in the csv file, it is often useful to look at the overlap between them. There are only 3 documents that are present in all 5 software (including [1]). This shows that there are differences between their algorithms. The table below expresses the partial overlap between the entries, i.e. an entry is counted as long as there is another entry within the scope of the criteria that identifies the same document.

| | Open Knowledge Maps | Connected Papers | Research Rabbit | Litmaps | Inciteful |
|---|---|---|---|---|---|
| \|documents\| | 53 | 41 | 50 | 20 | 31 |
| \|same documents wrt itself\| | 2 | 0 | 2 | 0 | 0 |
| \|same documents wrt others\| | 12 | 25 | 15 | 11 | 19 |
| % | 23% | 61% | 29% | 52% | 61% |

Table 5: Entries overlap across the 5 maps by software

The row with the % symbol in the last criterion is used to explain the percentage of each cell value compared to the the value in the $1^{rst}$ criterion.

$$\texttt{eg:} \quad \frac{12}{53} = 23\%$$

Surprisingly, OKM and RR both have 2 entries that are duplicated in their map. As explained in the Section 3.5, this is due to the source of OKM. Its source, Base search, allows different registered content providers to index the same document, which receives different identifiers. RR only mentions that it uses multiple sources which would allow the same problem to occur. Both have less overlap than the other three software, with respectively 53 and 50 documents referenced.

## 3.7 Map relations

Another way of assessing the quality of a map is to look at how the entries are connected to each other. The previous Section showed how much the abstracts of the entries were related, and this Section will see if the same tendencies can be observed visually. The table below selects some criteria for this:

| | Open Knowledge Maps | Connected Papers | Research Rabbit | Litmaps | Inciteful |
|---|---|---|---|---|---|
| \|documents\| | 53 | 41 | 50 | 20 | 31 |
| \|Single documents\| (no links) | 1 | 0 | 9 | 0 | 0 |
| \|documents linked to [1]\| | 10 | 27 | 13 | 19 | 5 |
| % | 19% | 66% | 26% | 95% | 16% |
| Clustering | Abstract similarity | No clusters | No clusters | Citation (Linear) | No clusters |
| Connections | Abstract similarity | Bibliographic Coupling & Co-citation | Citation networks and more | Citation network | Citation network |

Table 6: Statistics on the observable connections inside the maps by software

The use of the % symbol is the same as in the previous Section.

Only RR has the largest number of entries that don't show any visible link to other documents. For the third criterion, the software are very different. CP and LM have more than 60% of their documents directly linked to [1], while the others are all below 30%. For OKM this shows how close the metadata of the documents are,

whereas the software based on citation metrics show more the depth of their search within the citation network. Only OKM and LM cluster their entries according to a specific feature (see table 6). The last criterion expresses which data each software uses to find the connections between two documents. Unfortunately, no information is given on how they use this data.

## 3.8 Maps' representation

This Section evaluates the overall quality of the maps in terms of the visual aspect, which doesn't depend on the numerical results.

| | Open Knowledge Maps | Connected Papers | Research Rabbit | Litmaps | Inciteful |
|---|---|---|---|---|---|
| Document spreading | Spread across map | Centred documents | Centred documents | Spread across map | Spread across map |
| Connections | Explicit | Explicit | Explicit | Explicit | Explicit |
| Overlapping entities | Very few (<10) | Many (100«) | Some (<100) | Some (<100) | Very few (<10) |

Table 7: Maps qualitative comparison by software

Different choices can be seen across the software. Some software focused their maps around [1], producing a large 'cluster', while others gave more importance to exploration, resulting in a more spread out map. This focus has some correlation with the amount of overlapping entities they contain (an entity being defined as either a document or a visible link between two of them). For the same reason, there is also a correlation with the saturation of each map (how much space isn't used by any entity). Compared to the other software, CP has many more overlapping entities. This is because it shows any similarity with a line, the strength of the colour of which represents the value of the similarity. OKM is the only one that does not explicitly show its links. As explained in Chapter 2, this can be misleading, as the distance between two entries in different clusters doesn't represent their closeness.

## 3.9 Software features

Another important aspect of the software is their ease of use. How easily do the resulting maps tend to better redefine the queries? The features that enable this behaviour may not be a criterion for map quality, but they allow users to gradually adapt their queries to indirectly obtain better map quality. The next table helps the user to see which features are available for each of them:

| | Open Knowledge Maps | Connected Papers | Research Rabbit | Litmaps | Inciteful |
|---|---|---|---|---|---|
| User story 1 | NaN | Yes | Yes | Yes | Yes |
| User story 2 | NaN | Yes | Yes | Yes | Yes |
| User story 3 | NaN | No | Yes | Yes | No |
| User story 4 | NaN | No | No | No | Yes |
| User story 5 | NaN | No | Yes | No | No |

Table 8: Features by software

The user stories are:

1. multiple documents can be added as seeds for a map
2. a new map can be created from a document within the current map
3. the map axis can be selected
4. Various similarity criteria are available
5. The map authors, references, citations can be used to explore a topic of interest or to create a new map from suggestions.

The impact of each user story is discussed in the next Chapter.

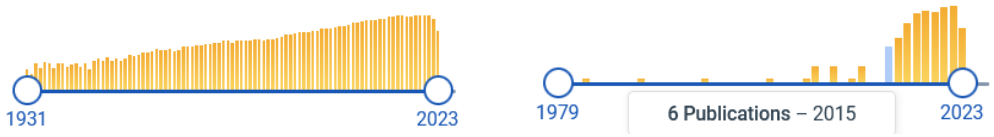
# 4 Discussion

The aim of this Chapter is to discuss the significance of the results of the Analysis and comparison Chapter. For this reason the structure of this Chapter will be identical to the previous one.

## 4.1 Entries' year of publication

In the following case study, both the seed document [1] for all citation based software studied in the Analysis and comparison Chapter and OKM's query are about the Travelling Salesman Problem (TSP) and drones.

The first mathematician to have written about TSP was K. Menger in the year of 1930 [35]. Since then the subject continuously gained popularity as shown on the following figure. There are almost 90 thousands documents available on Semantic Scholar when searching "Travelling salesman problem". Drones are a more recent topic (see Google trends in annexed FIGURE 20). TSP with drone (TSP-D) became researched around the same period as drones in 2015. TSP-D was first introduced by Murray and Chu in [36].




(a) Publications histogram on TSP (b) Publications histogram on TSP-D

Figure 13: Semantic Scholar publications histograms

The median of the year of publication for the documents harvested from RR's map is equal to 2016. It can be said that its search is more relaxed than the other software, since TSP-D was not mentioned before 2015 (see Section 3.2).

The other software's distributions mainly fall within the trend shown in FIGURE 13b. There are several reasons for these variations. The available documents depend on the data providers that each software uses. They do not all index the same documents. For the citation-based software, the used metrics have an impact on the timeliness of the documents. Logically co-citation metrics tend to favour older documents. Indeed, the later a document is, the more newer documents can reference it. On the other hand, bibliographic coupling is biased towards more recent documents. The more recent a document is, the more published documents can be referenced in it. An earlier median year of publication hints toward metrics that are more co-citation oriented.

Some variability is also due to the fact that data providers don't always agree on the year of publication. For example, the query "On the min-cost traveling salesman problem with drone" on Base Search returns 8 hits referring to the same document, but with 4 different publication dates. This is because there can be multiple versions of

a document, and that the document is not always made available online immediately after it has been released. Some data providers also send documents through different statuses: Received, Revised, Accepted, Available. Data providers may have different speeds to go through their process, which influences the results of a query.

## 4.2 Entries' citation counts

For the same reason as in the previous Section, the differences between the software will be more important than the general trends, which may only be relevant for this use case. Citations can be modelled as a monotonic increasing function[2] over time, i.e. each document can only gain citations over time. This dependency means that the year of publication also plays a role in this Section. In fact, RR, which has a median year of publication far away from the other software, also has a much higher mean number of citations.

Conversely, OKM returns a lower mean. As mentioned in Section 2.3.1 OKM focuses only on the textual metadata provided by Base Search. The documents identified in OKM's map are not always referenced by the other data providers, hence the large number of documents with 0 citation counts.

The frequency of 0 citation counts is also a concern for citation-based software. The bibliographic coupling metric identifies documents that cite the same set of documents, although there is no requirement for their own citation counts. LM and ICT 0 count frequencies suggest the use of this metric. As suggested in the previous Section on bibliographic coupling, LM has a high median publication date. ICF doesn't, but older documents that are not cited may still be affected by this metric.

As in the previous Section, the accuracy of citation metadata depends on the consistency of the data provider. A document citation count may not be up to date and have a lower count, or it may not take into account citations from documents that are not indexed by that data provider.

A recommendation from this thesis for this Section would be to retrieve the citation network of the relevant documents to perform a network analysis and compare the results with the citation based software.

## 4.3 Entries' abstracts' similarity

Using the TF-iDF approach to calculate the similarity between two document abstracts isn't without its drawbacks. It consists of counting the frequency of every static word embedding (i.e. lemma in this case). The problem is that lemma correlations may be overlooked, thus complicating the calculation.

There are more advanced approaches to this problem, such as using contextual embedding [38]. This dynamically computes the embedding of words (influenced

[2] "A monotonic function is a function which is either entirely non-increasing or non-decreasing." - [37]

by the context in which the word is found). This approach could be explored to reduce correlation inaccuracies. Contextualised embedding wasn't used in this thesis because it requires texts that are much longer than the document abstracts.

The TF-iDF approach, being a text-based approach, makes OKM stand out as it is the only text-based software studied in Chapter 3. The other software performances (as seen in FIGURE 12) show the correlation between the text-based and citation metrics. CP's performance is reduced because it truncates abstracts if they are too long.

## 4.4 Data correctness

As previously mentioned, the data provider plays an important role in the consistency of the data. These inaccuracies could lead to different results using the same algorithms but different data providers, or several of them at the same time.

Text metadata are also susceptible to these risks. Each data provider and software uses its own algorithm to retrieve the metadata of the documents to be exploited. Many different scenarios have been observed when harvesting the metadata of documents placed in the csv file constructed in the introduction of Chapter 3. Some of these problems were: not being able to read the character encoding system of a document resulting in the transformation of the original characters (mostly applicable to special characters), wrong classification of parts of the document (author included in the abstract field of the metadata), partial content not retrieved due to size constraints, etc. All this will affect the results.

For this Section, it is very difficult to determine which software is correct when an inconsistency is found. Therefore, no conclusions can be drawn about the software, except for what was reported in Section 4.1 on base searching, where different data providers may index the same documents with different metadata.

## 4.5 Documents' overlap

Overlapping within a single map was explained in the previous Section. The fact that a single document can be retrieved multiple times from the same or different data providers, and sometimes with inconsistent metadata, can lead to duplicate documents if a software doesn't check for these errors.

The overlap between different maps is heavily influenced by the choice of data providers. Since they do not index the same documents, the overlap between two sets of documents is limited.

The software approaches (text-based or citation-based) will also play an important role in this Section. Using different data sets increases the chances of divergence. In fact, OKM doesn't have much overlap with the other software. This is due to its choice of Base Search as data provider (the others have different and some common data providers) and because it is the only text-based software in the selection made in the Analysis and comparison Chapter.

RR would be expected to have a higher overlap with the other citation-based software on this basis, but the difference in year of publication means that a significant proportion of its map documents are older and therefore different from those of the other software.

## 4.6 Map relations

Regarding the mapping algorithms, nothing can be said about single documents (i.e. no links with other documents), as they are placed in such a way to saturate the map as little as possible. The greater the proportion of single documents in a map, the less knowledge there is in that map. RR's map contains 9 single documents, whose their presence is left to the user's speculation.

The set of documents directly linked to an input document (seed document) represents the depth of the search. A too large set suggests a shallow search, and a too small set suggests a deep search, as long as the links still represent the "cited by" relationship. RR and LM apply this relationship logic. This means that LM performs a shallow search while RR performs a deep search. On the other hand, CP and ICF don't apply the "cited by" relationship, so nothing can be deduced from this. For OKM, there was no actual seed document, but the target document, which was used as seed document for the other software, still remains in the largest cluster of its map, which means that the input query is of good quality.

The clustering algorithms try to make it easier to explore a map by sorting it thematically. For the LM map, using the clustering axis (optional) provides a map with less overlap and more dispersed documents. For OKM, the map is organised into subdomains, which helps to target specific documents.

## 4.7 Maps' representation

This Section concerns the readability of each map. Both the number of overlapping entities and their distribution on the maps are obstacles to getting a good overview of the documents present in each map. The fact that CP does both of them makes it more difficult to read, although the entities are highlighted when selected. OKM clustering from the previous Section makes it very easy to navigate through its map. Another advantage of OKM is that documents are identified by their title rather than by their main author, as in all the other software. However, thanks to its mapping algorithms, no maps have overlapping documents.

## 4.8 Software features

This Section will talk about the impact of each of the user stories mentioned in Section 3.9.

The first user story allows for flexibility. The user can try to find the links between different documents by creating maps. It could help to fill in the gaps between different researchers, authors, institutions, etc.

The second user story is simply to save the user time, as a new map can always be created from scratch by modifying the query of the old map.

The third user story query helps the user to customise the maps. In this way, some information from the maps can be displayed and made more obvious (identification of clusters, relationships between authors, contribution of institutions, etc.).

The fourth user story offers different criteria and therefore more varied documents as possible paths to orient user research.

The fifth user story provides other possible ways of exploring a domain of interest and can be compared with the metrics used in the software studied in this thesis to see the extent to which these graphs provide relevant and useful information.

# 5 Conclusion

The current research aimed to investigate some of the most advanced literature dedicated visual mapping software. They are divided into two groups: (i) those that generate knowledge maps and (ii) those that generate knowledge graphs. The software in the first group are Open Knowledge Maps (OKM), Connected Papers (CP), Litmaps (LM), Inciteful (ICF) and Research Rabbit (RR). The software in the second group are Open Research Knowledge Graph (ORKG) and InfraNodus.

The central objectives were as follows:

1. Present an overview of current advances in the field of visual mapping through a survey

2. Compare and evaluate the software of the first group (i) through a case study

The case study consists of generating a map around the same document for all the selected software. The document that was agreed upon during the preparation of the thesis was [1]. This document treats the travelling salesman problem with drone (TSP-D). ORKG and InfraNodus will not be part of the evaluation because they do not really solve the problem identified in the introduction, which is to help users get a good overview of the literature on a topic of interest. For the only studied software that uses text queries, OKM, the input was "travelling salesman problem drone". For the remaining software, requiring documents as input (seed documents), [1] was entered as the only seed document.

The result of the research demonstrates that although all the software provide relevant maps and graphs, their workflow can be very different. OKM and RR provide more diverse documents than the other software. RR's map suggests a more relaxed implementation that searches and selects documents further away from its seed document. OKM's choice to use the textual metadata of the documents, as well as its different choice of data providers, makes it search and select documents from different libraries than the other software. On the other hand, CP, LM and ICF have more overlap as they have more documents in common between their maps.

Although their workflows are different, all software show signs of data inconsistency. Document metadata were found to have been altered during extraction, sometimes deliberately. Some data were missing, either because the software could not find them or because they were missing deliberately. Some data were out of date. Some data were different due to lack of conventions. Duplicate data were also found. These inconsistencies often come from the data providers and are overlooked by any software. The rest of the time the software are directly responsible for changing the data.

The ease of use of the software also plays an important role in the evaluation. The first aspect is visual. OKM's clustering gives a clear view of the domain outlined by its map, whereas CP's choice of connecting lines between documents has the opposite effect, flooding its map, which makes it difficult to read. The second aspect

is the feasibility or ease with which a software helps the user to explore a domain. RR stands out for its ability to iteratively refocus the user's current query. Although OKM doesn't offer much flexibility once a map has been generated, its query format offers many parameters to compensate for this.

The limitation of this work is that the evaluation was only based on a single case study because of time constraints. The TSP-D is a relatively new topic (since 2015) and not one of the most researched. The question is whether the same analysis would work as well with other seed documents. Each domain has its own citation network tree, its own document correlations, etc. More study cases would be needed to confirm the initial results obtained from the TSP-D query. Hence it would be particularly interesting to test the software with multiple seed documents, dealing with topics that have been researched over a longer period of time, with many citations. More recent topics with fewer citations might also yield interesting results.

Based on these conclusions, users should consider using more than one software to ensure they get the best of all worlds. They should always bear in mind that some data may be truncated, altered or missing and that a double check is required before using them.

# Bibliography


[1] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "On the min-cost traveling salesman problem with drone," *ArXiv*, vol. abs/1512.01503, 2015.

[2] P. Kraker, C. Kittel, and A. Enkhbayar, "Open knowledge maps: Creating a visual interface to the world's scientific knowledge based on natural language processing," *027.7 Zeitschrift für Bibliothekskultur*, vol. 4, no. 2, pp. 98–103, Nov. 2016. [Online]. Available: https://doi.org/10.5281/zenodo.4705327

[3] A. S. Foundation. Package org.apache.lucene.search - scoring. (Accessed: 29-09-2022). [Online]. Available: https://lucene.apache.org/core/6_4_2/core/org/apache/lucene/search/package-summary.html#scoring

[4] A. Jivani, "A comparative study of stemming algorithms," *Int. J. Comp. Tech. Appl.*, vol. 2, pp. 1930–1938, 11 2011.

[5] L. Leydesdorff, "Similarity measures, author cocitation analysis, and information theory," *J. Assoc. Inf. Sci. Technol.*, vol. 56, pp. 769–772, 2005.

[6] L. Egghe, "Good properties of similarity measures and their complementarity," *J. Assoc. Inf. Sci. Technol.*, vol. 61, pp. 2151–2160, 2010.

[7] A. W. Mead, "Review of the development of multidimensional scaling methods," *The Statistician*, vol. 41, pp. 27–39, 1992.

[8] P. Kraker, C. Schlögl, K. Jack, and S. Lindstaedt, "Visualization of co-readership patterns from an online reference management system," *Journal of Informetrics*, vol. 9, no. 1, pp. 169–182, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1751157714001151

[9] Zemanta. Zemanta api. (Accessed: 05-10-2022). [Online]. Available: http://dev.zemanta.com/one/api/

[10] Refinitiv. Intelligent tagging & text analytics. (Accessed: 05-10-2022). [Online]. Available: http://opencalais.com/

[11] J. E. Ramos, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, 2003.

[12] S. Auer, "Towards an open research knowledge graph," Jan. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1157185

[13] S. Auer, A. Oelen, M. Haris, M. Stocker, J. D'Souza, K. E. Farfar, L. Vogt, M. Prinz, V. Wiens, and M. Y. Jaradeh, "Improving access to scientific literature with knowledge graphs," *Bibliothek Forschung und Praxis*, vol. 44, pp. 516 – 529, 2020.

[14] P. Lopez, "Grobid: Combining automatic bibliographic data recognition and term extraction for scholarship publications," in *Research and Advanced Technology for Digital Libraries*, M. Agosti, J. Borbinha, S. Kapidakis, C. Papatheodorou, and G. Tsakonas, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 473–474.

[15] A. Oelen, M. Stocker, and S. Auer, "Creating a scholarly knowledge graph from survey article tables," in *Digital Libraries at Times of Massive Societal Transition*, E. Ishita, N. L. S. Pang, and L. Zhou, Eds. Cham: Springer International Publishing, 2020, pp. 373–389.

[16] A. S. Corrêa and P.-O. Zander, "Unleashing tabular content to open data: A survey on pdf table extraction methods and tools," in *Proceedings of the 18th Annual International Conference on Digital Government Research*, ser. dg.o '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 54–63. [Online]. Available: https://doi.org/10.1145/3085228.3085278

[17] M. Y. Jaradeh, A. Oelen, K. E. Farfar, M. Prinz, J. D'Souza, G. Kismihók, M. Stocker, and S. Auer, "Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge," in *Proceedings of the 10th International Conference on Knowledge Capture*, ser. K-CAP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 243–246. [Online]. Available: https://doi.org/10.1145/3360901.3364435

[18] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2016.

[19] D. Paranyushkin, "Infranodus: Generating insight using text network analysis," in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 3584–3589. [Online]. Available: https://doi.org/10.1145/3308558.3314123

[20] ——. Text network analysis and data visualization tutorials. (Accessed: 07-11-2022). [Online]. Available: https://infranodus.com/docs

[21] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977. [Online]. Available: http://www.jstor.org/stable/3033543

[22] S. Fortunato, "Community detection in graphs," *ArXiv*, vol. abs/0906.0612, 2009.

[23] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, oct 2008. [Online]. Available: https://dx.doi.org/10.1088/1742-5468/2008/10/P10008

[24] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software," *PLoS ONE*, vol. 9, 2014.

[25] N. Labs. Publications. (Accessed: 07-11-2022). [Online]. Available: https://noduslabs.com/publications/

[26] H. G. Small, "Co-citation in the scientific literature: A new measure of the relationship between two documents," *J. Am. Soc. Inf. Sci.*, vol. 24, pp. 265–269, 1973.

[27] M. M. Kessler, "Bibliographic coupling between scientific papers," *American Documentation*, vol. 14, pp. 10–25, 1963.

[28] K. W. Boyack and R. Klavans, "Co-citation analysis, bibliographic coupling, and direct citation: Which citation approach represents the research front most accurately?" *J. Assoc. Inf. Sci. Technol.*, vol. 61, pp. 2389–2404, 2010.

[29] E. Smolyansky. Announcing connected papers — a visual tool for researchers to find and explore academic papers. (Accessed: 02-01-2023). [Online]. Available: https://medium.com/connectedpapers/announcing-connected-papers-a-visual-tool-for-researchers-to-find-and-explore-academic-papers-89146a54c7d4

[30] Hamish. Guide to litmaps visualisations. (Accessed: 04-01-2023). [Online]. Available: https://medium.com/litmaps/guide-to-litmaps-visualisations-95a9bc2cc9de

[31] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking : Bringing order to the web," in *The Web Conference*, 1999.

[32] D. Liben-Nowell and J. M. Kleinberg, "The link-prediction problem for social networks," *J. Assoc. Inf. Sci. Technol.*, vol. 58, pp. 1019–1031, 2007.

[33] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Soc. Networks*, vol. 25, pp. 211–230, 2003.

[34] R. R. team. Welcome to the faq! (Accessed: 6-01-2023). [Online]. Available: https://researchrabbit.notion.site/Welcome-to-the-FAQ-c33b4a61e453431482015e27e8af40d5#cda19af8e44f46e6a11f972a4c3eacd5

[35] A. Schrijver, "On the history of combinatorial optimization (till 1960)," in *Discrete Optimization*, ser. Handbooks in Operations Research and Management Science, K. Aardal, G. Nemhauser, and R. Weismantel, Eds. Elsevier, 2005, vol. 12, pp. 1–68. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705120015

[36] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C-emerging Technologies*, vol. 54, pp. 86–109, 2015.

[37] C. Stover. Monotonic function. (Accessed: 02-04-2023). [Online]. Available: https://mathworld.wolfram.com/MonotonicFunction.html

[38] A. Akbik, D. A. J. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *International Conference on Computational Linguistics*, 2018.

# Appendix

## InfraNodus workflow

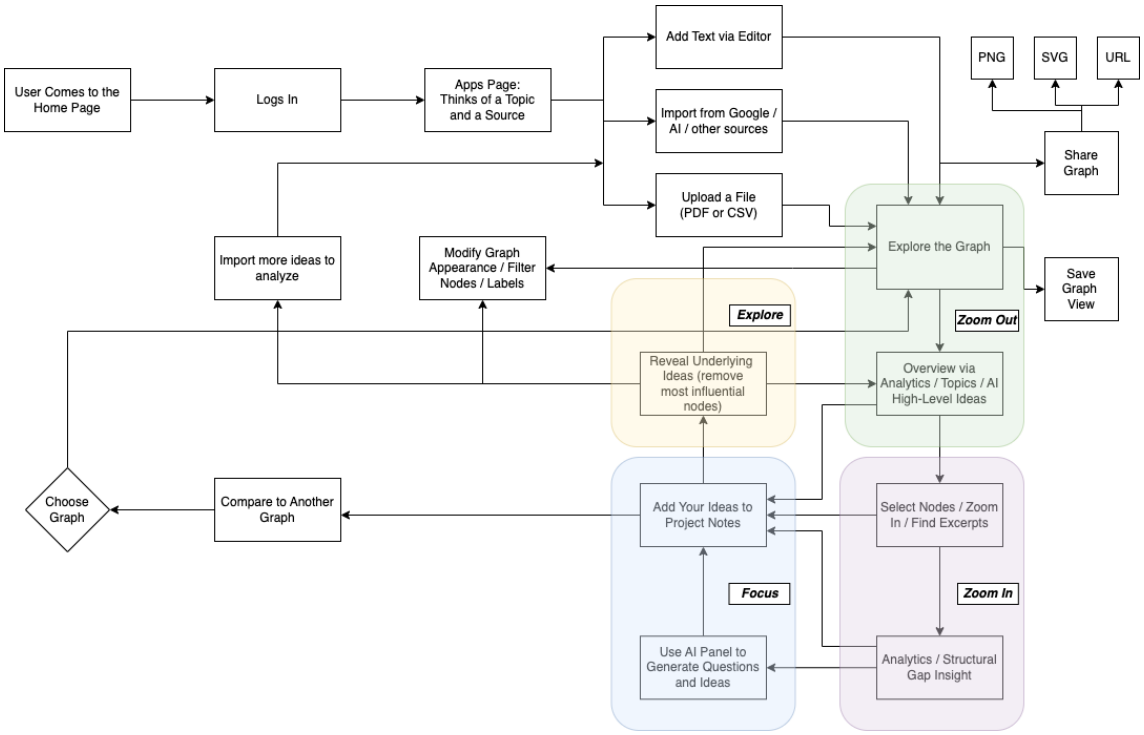



Figure 14: InfraNodus ecological thinking workflow

## Maps

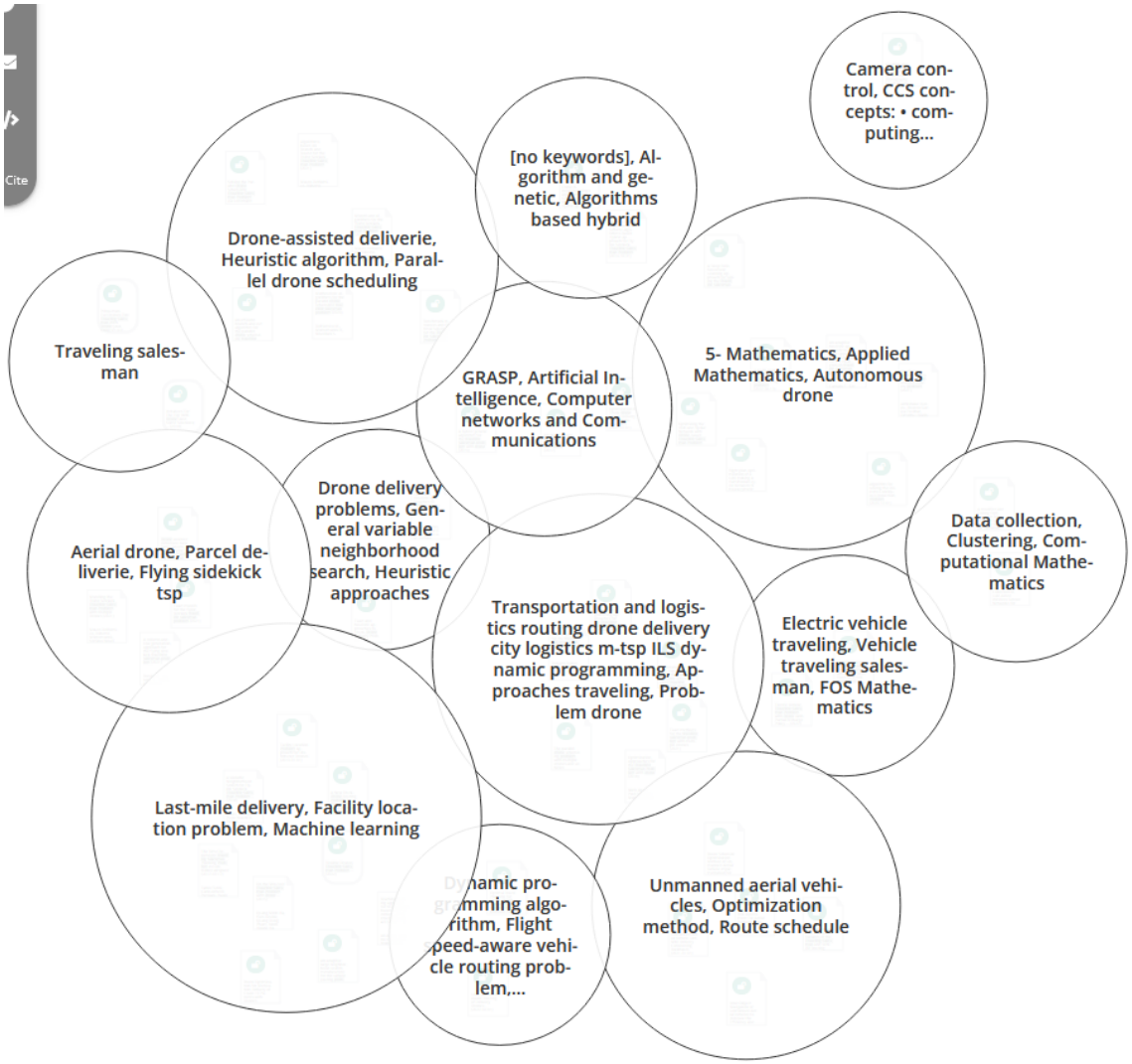



Figure 15: Open Knowledge Maps' map

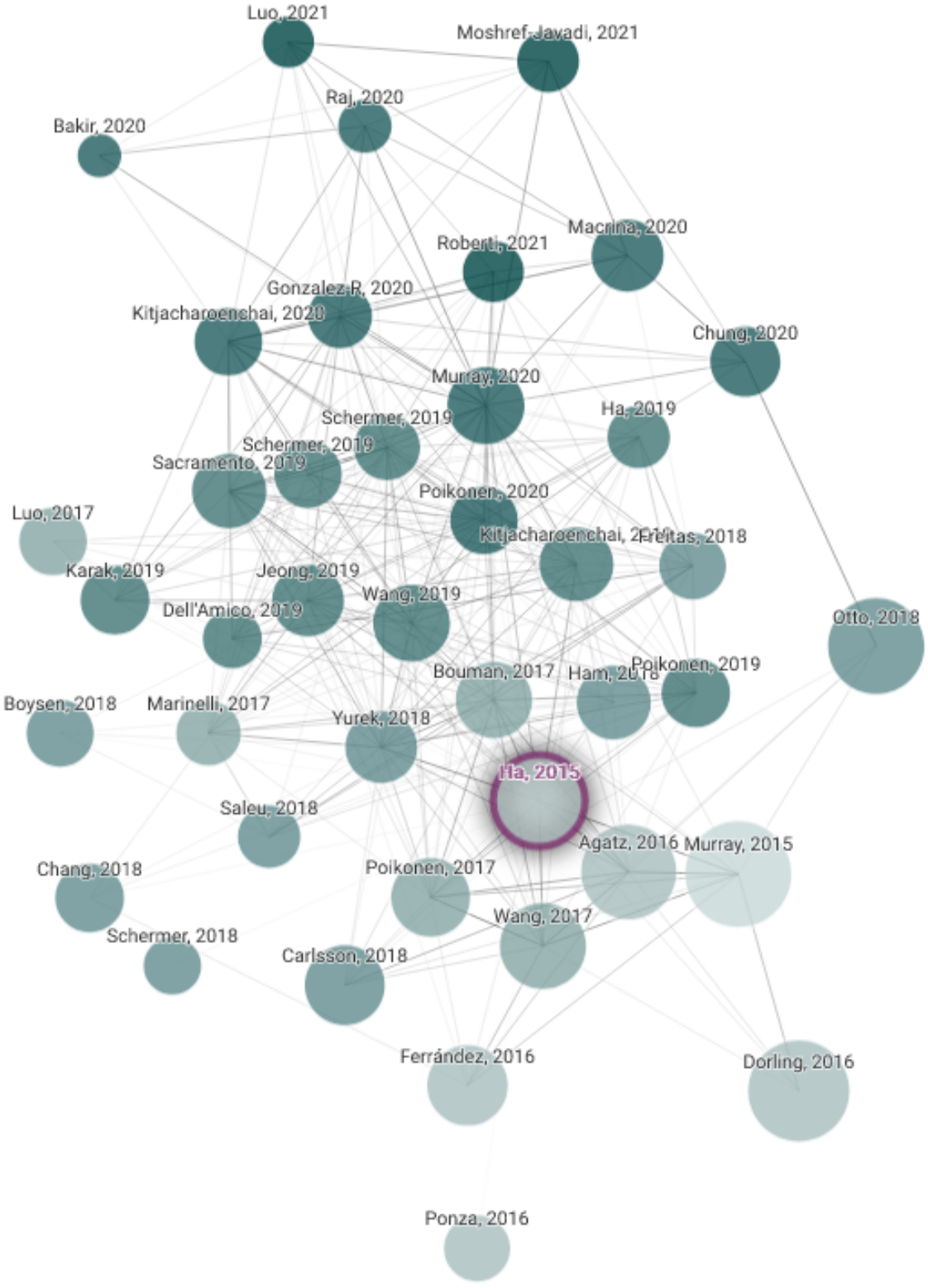



Figure 16: Connected Papers' map

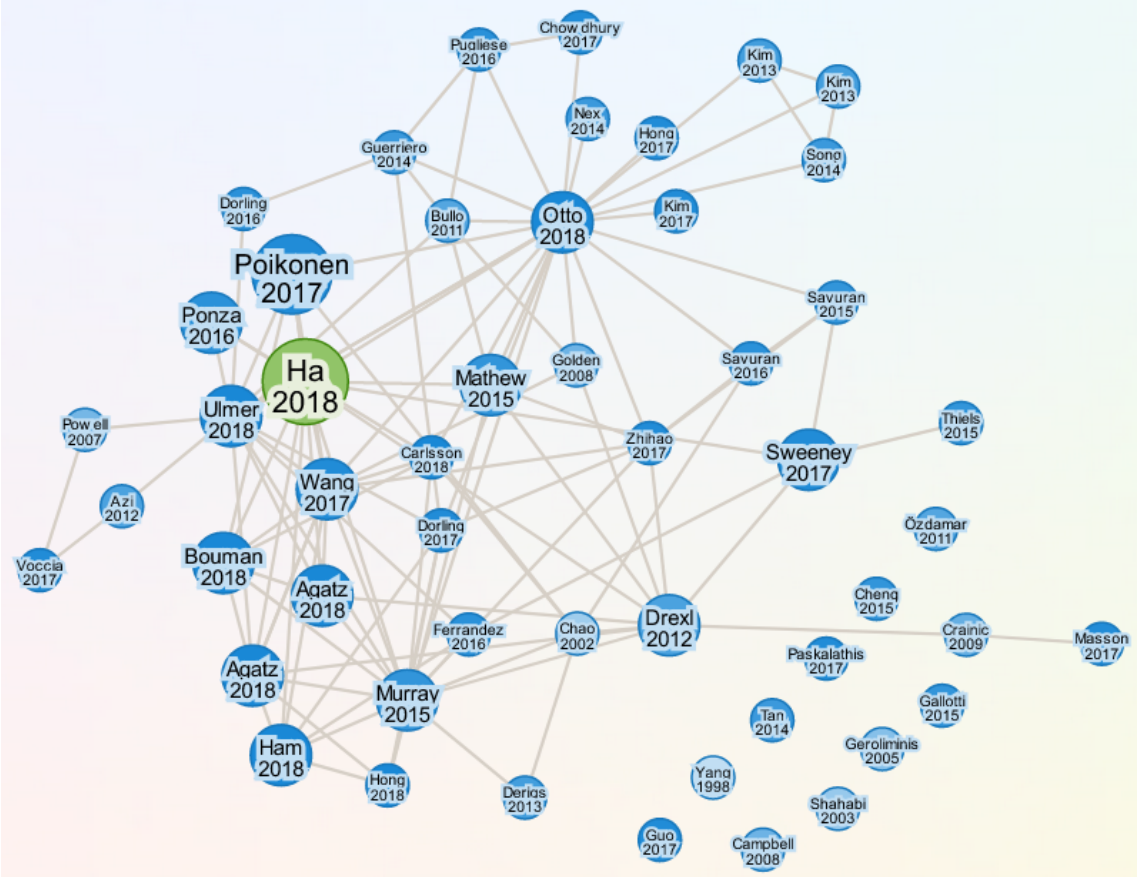



Figure 17: Research rabbit's map

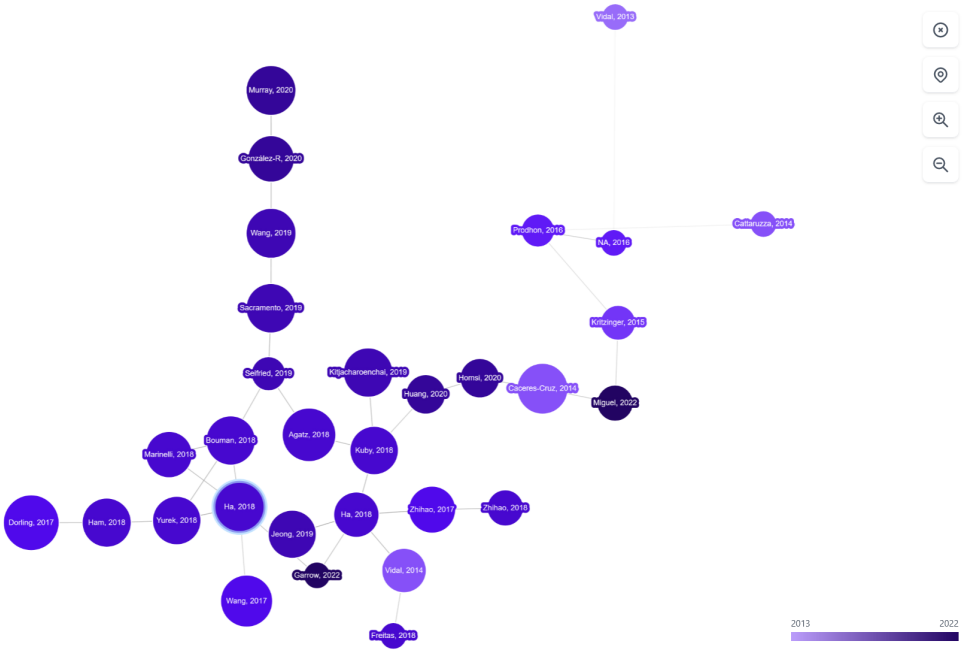



Figure 18: Inciteful's map

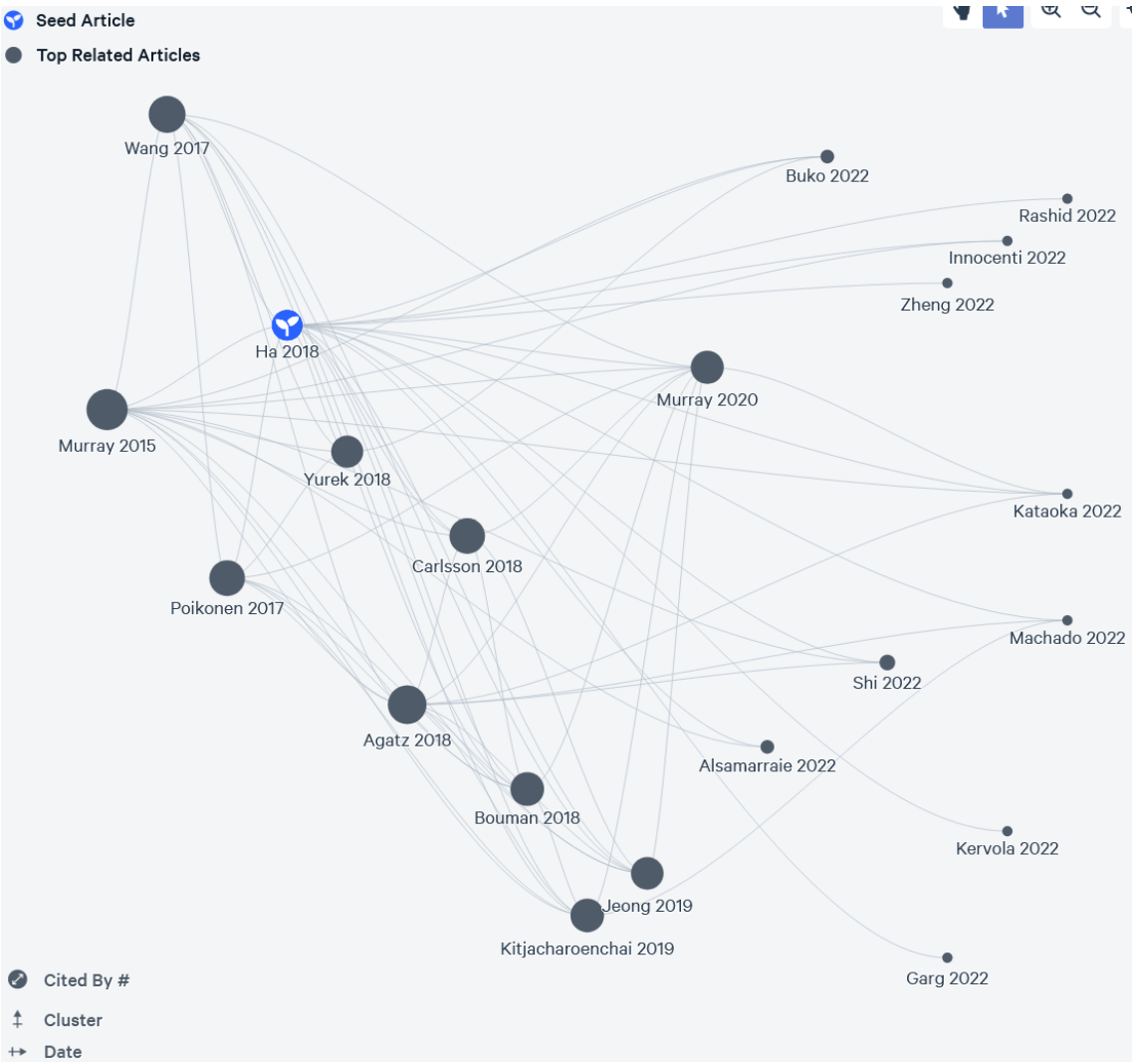



Figure 19: Litmap's map

## Drone on Google Trends

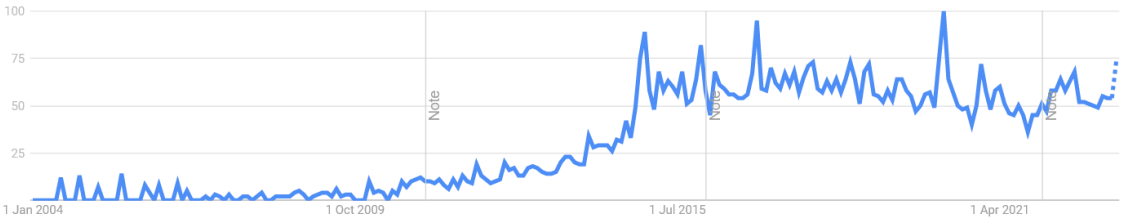



Figure 20: Drone on Google Trends

# Code

## Import

```
import numpy as np
import pandas as pd

df = pd.read_csv('tester.csv', sep=',')

df['Title_lower'] = df['Title'].astype(str).str.lower()

plus = pd.read_csv('okm.csv', sep=',',header=None)
complete = df.copy(deep=True)
complete.Description.fillna('', inplace=True)

for i in range(53):
    complete.loc[i,'Citations'] =  plus[1][i]

#complete.info()
```

## Lemmatizer

```
from nltk import word_tokenize, pos_tag
from lemminflect import getLemma, getAllLemmas, getAllLemmasOOV
from nltk.lm import Vocabulary
from sklearn.feature_extraction.text import TfidfTransformer, CountVectorizer
from sklearn.pipeline import Pipeline


matrix = []
for index, row in df.iterrows():
    if type(row['Description'])==float:
        matrix.append('')
        continue

    desc = ' '.join([j.lower()
                     for j in [i.strip('.,:;!?-\'\"()[]«»%')
                               for i in row['Description'].split()] if j!=''])

    tokens = pos_tag(word_tokenize(desc), tagset='universal')
    my_list = []
    for word, tag in tokens:
        lemma = getLemma(word, upos=tag)
        if len(lemma)==0: continue
        my_list.append(lemma[0])

    matrix.append(' '.join(my_list))

corpus = matrix
vocabulary = Vocabulary((' '.join(corpus)).split())
```

```
pipe = Pipeline([('count', CountVectorizer(vocabulary=vocabulary)),
                 ('tfid', TfidfTransformer())]).fit(corpus)
```

### TD-iDF

```
tfidf = (pipe['count'].transform(corpus).toarray() * pipe['tfid'].idf_)

from numpy.linalg import norm
from sklearn.metrics.pairwise import cosine_similarity

sim_matrix = []
for i in range(len(matrix)):
    l = []
    for j in range(len(matrix)):
        s = cosine_similarity(tfidf[i].reshape(1, -1),tfidf[j].reshape(1, -1))
        l.append((j, s[0][0]))
    sim_matrix.append(sorted(l, key=lambda x: x[1]))
```

### Data

```
groups = complete.groupby('Tool')
```

```
groups.size()
```

```
Tool
CP     41
I      31
L      21
OKM    53
RR     51
dtype: int64
```

```
print(complete['Citations'].sum())
groups.sum()
```

```
26166.0
```

```
      Citations    Year
Tool
CP       6512.0   82754
I        3193.0   62552
L        2220.0   42417
OKM      2228.0  107067
RR      12013.0  102717
```

```
groups.median()
```

```
      Citations    Year
Tool
CP        103.0  2019.0
I          72.0  2018.0
L          85.0  2020.0
OKM         9.0  2021.0
RR        110.0  2016.0
```

```
groups.mean()
```

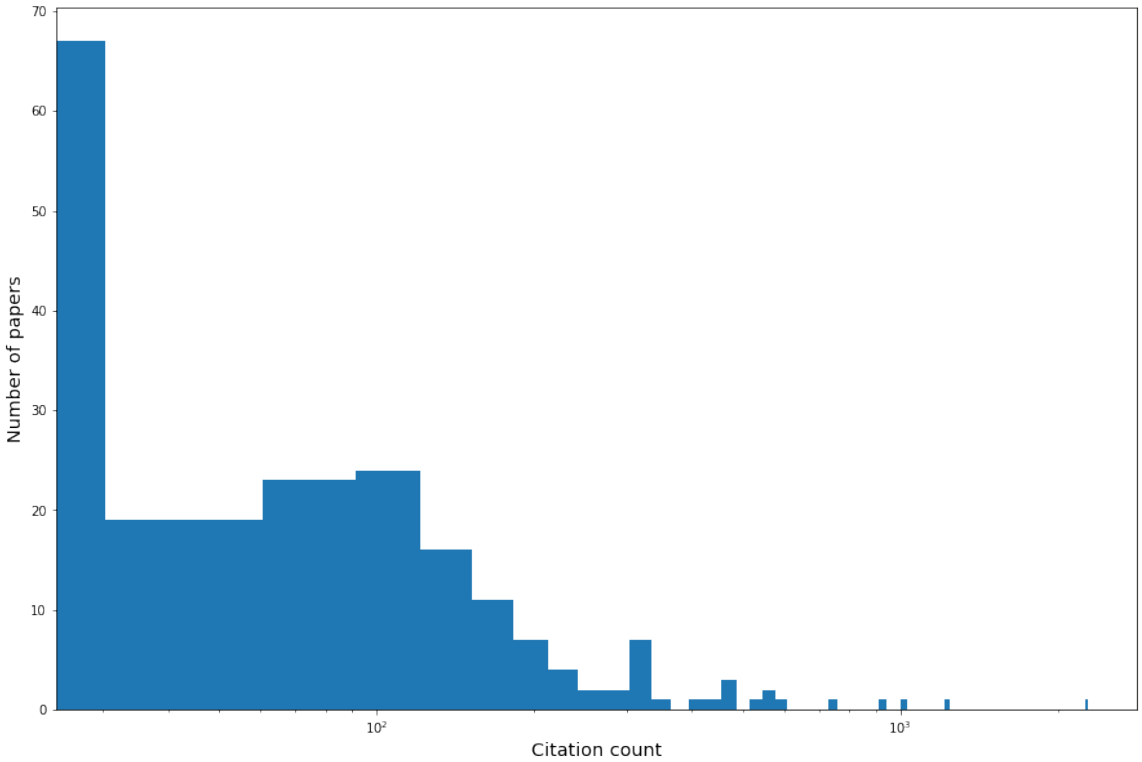
```
       Citations         Year
Tool
CP    158.829268  2018.390244
I     103.000000  2017.806452
L     105.714286  2019.857143
OKM    42.037736  2020.132075
RR    235.549020  2014.058824
```

```
groups.std()
```

```
       Citations      Year
Tool
CP    154.302771  1.594962
I     124.836426  2.197261
L     141.585360  2.286607
OKM    89.867202  1.808503
RR    382.524944  4.593089
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))
complete['Citations'].plot.hist(grid=False, bins=75)
plt.xscale('log')
plt.ylabel('Number of papers', size='x-large')
plt.xlabel('Citation count', size= 'x-large')
```

```
Text(0.5, 0, 'Citation count')
```




```
tot = 50
```

```
size = len(complete)
bol = True
while(bol):
    tot+=1
    bol = len(complete.loc[complete['Citations']>tot])/size>1/3

cited = complete.loc[complete['Citations']>tot]
print('#citations threshold :',
      tot,'--> #papers :',len(complete.loc[complete['Citations']>tot]))
#cited.sort_values(by=['Citations'], ascending=False)
cited.groupby('Tool').size()
```

```
#citations threshold : 115 --> #papers : 65
```

```
Tool
CP     18
I      11
L       7
OKM     5
RR     24
dtype: int64
```

```
print(len(df['Title']))
print(len(df['Title_lower'].drop_duplicates()))
print(len(df.loc[df.duplicated(subset=['Title'], keep=False), :]))
print(len(df.loc[df.duplicated(subset=['Title_lower'], keep=False), :]))
print(len(df.loc[df.duplicated(subset=['Title_lower','Citations'], keep=False), :]))
print(len(df.loc[df.duplicated(subset=['Title_lower','Description'], keep=False), :]))
print(len(df.loc[df.duplicated(subset=['Title_lower','Year'], keep=False), :]))
```

```
197
143
79
82
4
30
74
```

```
duplicates = df.loc[df.duplicated(subset=['Title_lower', 'Tool'], keep=False), :]
print(duplicates.groupby('Tool').size()/2)
title_set = set(duplicates['Title_lower'].tolist())
```

```
Tool
OKM    2.0
RR     2.0
dtype: float64
```

```
duplicates2 = df.loc[df.Title_lower.duplicated(keep=False), :]
print(duplicates2.groupby('Tool').size())
```

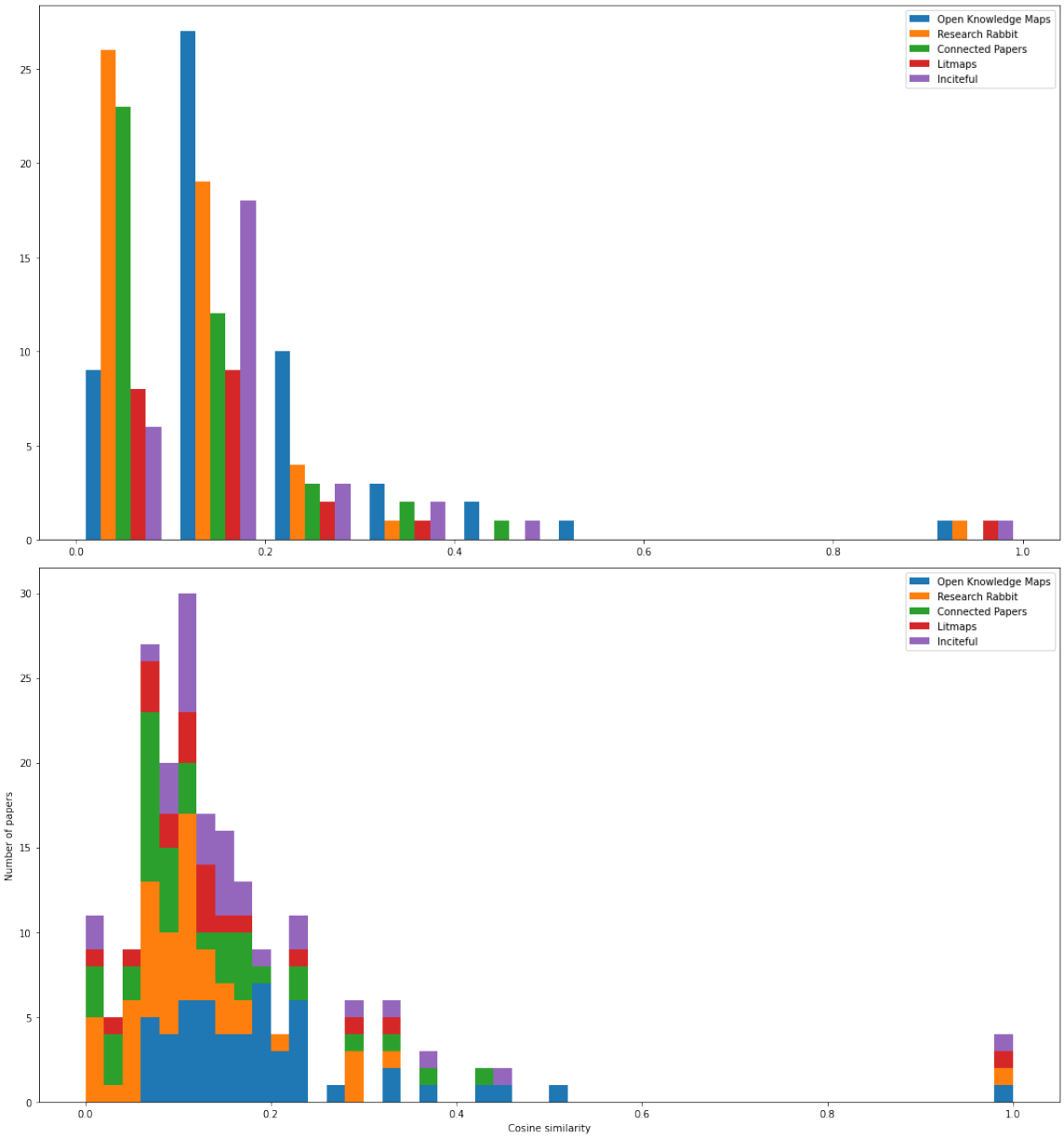
```
Tool
CP     25
I      19
L      11
OKM    12
RR     15
```

```
dtype: int64
```

```
print(df.index[df['Title'] == "On the Min-cost Traveling Salesman Problem with Drone"].tolist())
```

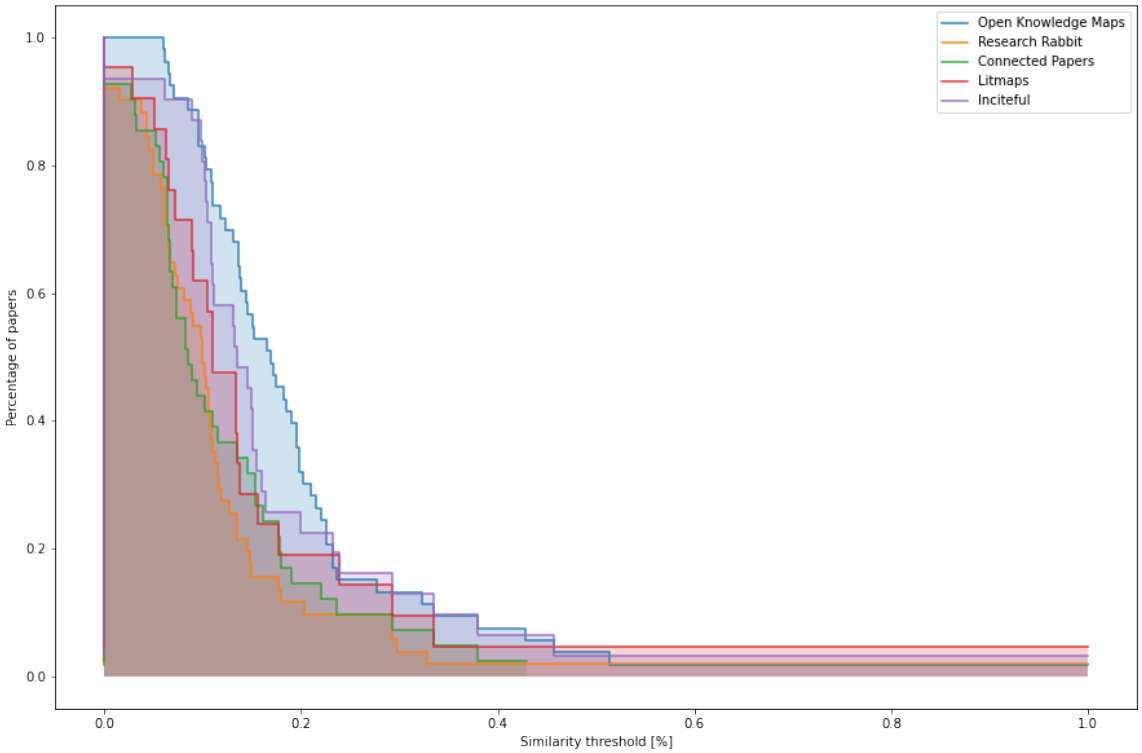
```
[15, 21, 53, 104, 145, 185]
```

```
import matplotlib.pyplot as plt
#df.index[df['Tool']=='I']
#OKM (0..52); RR (53..103); CP (104..144); L (145..165); I (166..196)
labels = df.Tool.unique().tolist()
full_labels = ['Open Knowledge Maps','Research Rabbit',
               'Connected Papers','Litmaps','Inciteful']

n_bins = 50
xs = [[],[],[],[],[]]

for i, j in sim_matrix[185]:
    if i<53:
        xs[0].append(j)
    elif i<104:
        xs[1].append(j)
    elif i<145:
        xs[2].append(j)
    elif i<166:
        xs[3].append(j)
    elif i<197:
        xs[4].append(j)

fig, ((ax0), (ax1)) = plt.subplots(nrows=2, ncols=1, figsize=(15, 16))

ax0.hist(xs, 10, histtype='bar', label=full_labels)
ax0.legend(prop={'size': 10})
plt.xlabel("Cosine similarity")
plt.ylabel("Number of papers")
plt.title("")

ax1.hist(xs, n_bins, histtype='bar', stacked=True, label=full_labels)
ax1.legend(prop={'size': 10})
plt.xlabel("Cosine similarity")
plt.ylabel("Number of papers")
plt.title("")


fig.tight_layout()
plt.show()
```



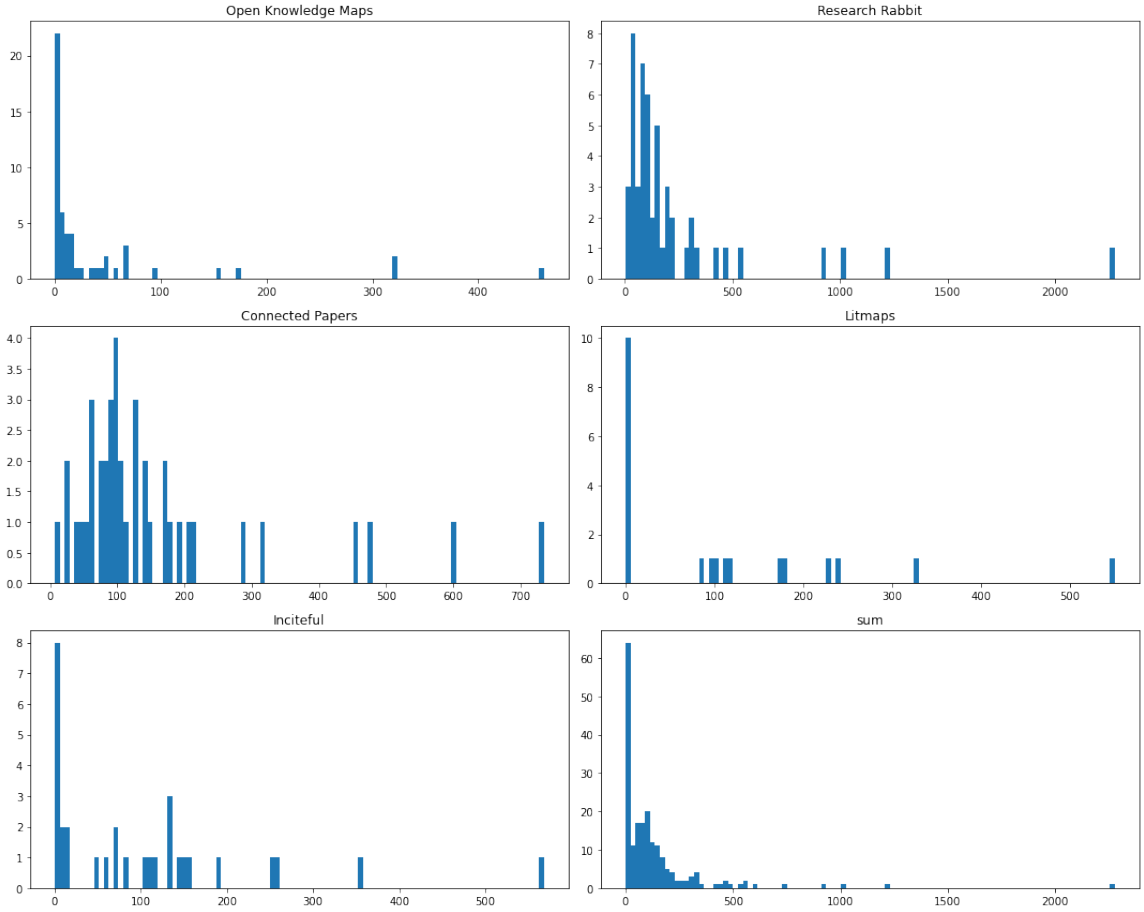

```
plt.figure(num = 3, figsize=(15, 10))
order = ['OKM','RR','CP','L','I']

for i in range(len(xs)):
    x = xs[i].copy()
    x.insert(0,0)
    #x = np.array(x) / groups.size()[order[i]]
    y = list(range(len(x),0,-1))
    y = np.array(y) / groups.size()[order[i]]
    y[0] -=1
    #plt.stairs(range(len(x)-1,0,-1), x, baseline=0, fill=True)
    plt.fill_between(x,y, step="pre", alpha=0.2)
```

```
    plt.plot(x, y,drawstyle="steps", alpha=0.9)

#OKM (0..52); RR (53..103); CP (104..144); L (145..165); I (166..196)
plt.legend(full_labels)
plt.xlabel("Similarity threshold [%]")
plt.ylabel("Percentage of papers")
#plt.xscale('log')
plt.title("")
plt.show()
```

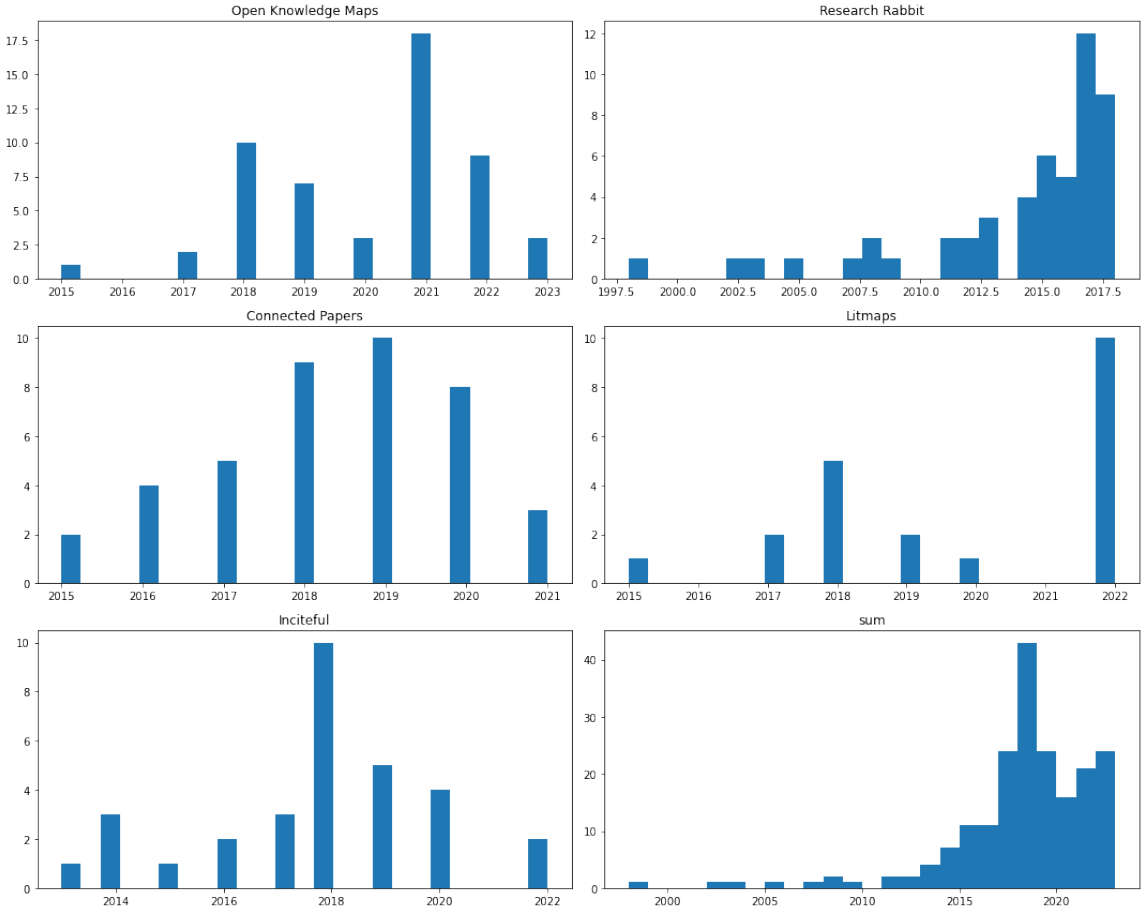



```
def histo(graph,column,tool,name, bins):
    x = complete[complete['Tool']==tool][column]
    graph.hist(x, bins, histtype='bar')
    graph.set_title(name)

def attribute(column, bins):
    figs = [ax0, ax1, ax2, ax3, ax4]
    for i in range(len(figs)):
        histo(figs[i], column, labels[i], full_labels[i], bins)

    ax5.hist(complete[column], bins, histtype='bar')
    ax5.set_title('sum')

    fig.tight_layout()
    plt.show()

fig, ((ax0, ax1), (ax2, ax3), (ax4, ax5)) = plt.subplots(nrows=3, ncols=2, figsize=(15, 12))
```

```
attribute('Citations', 100)
```




```
fig, ((ax0, ax1), (ax2, ax3), (ax4, ax5)) = plt.subplots(nrows=3, ncols=2, figsize=(15, 12))
attribute('Year', 25)
```



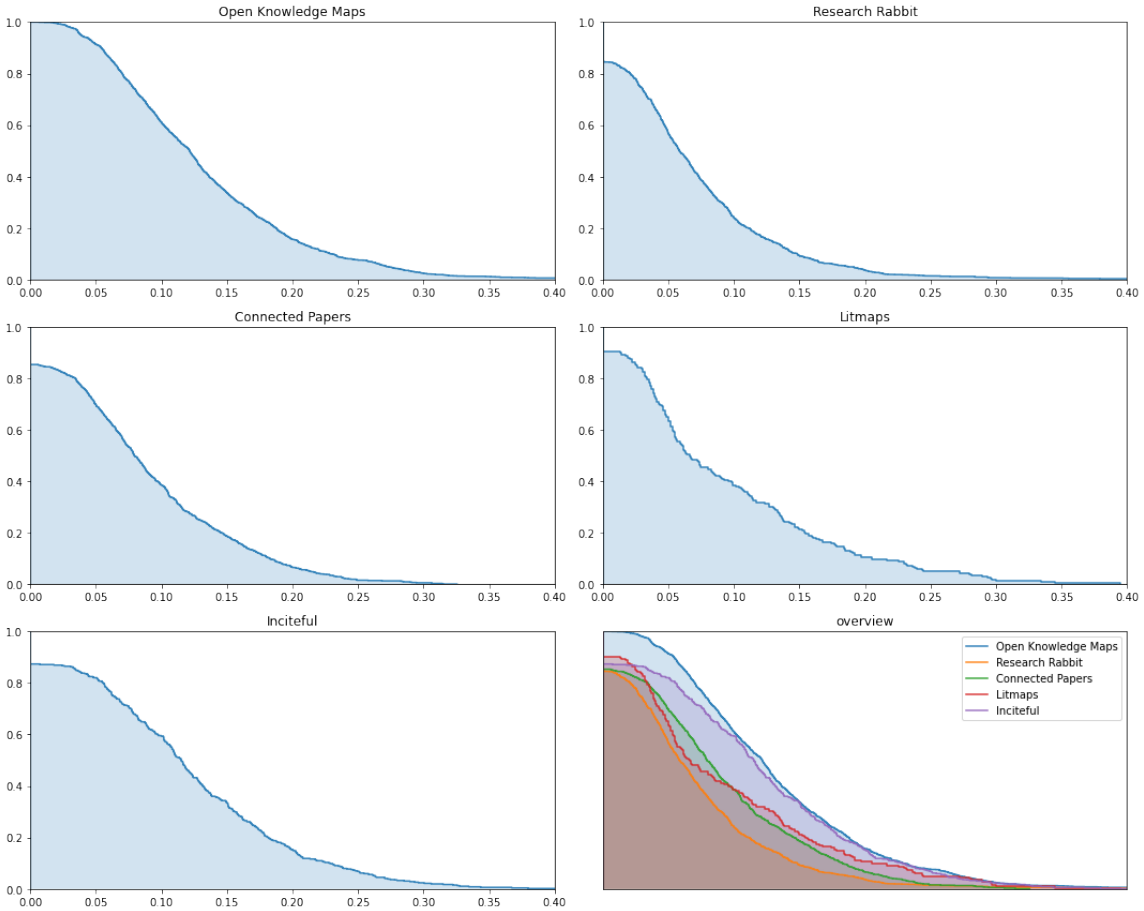

```
#OKM (0..52); RR (53..103); CP (104..144); L (145..165); I (166..196)
def getRange(index):
    if index<53: return 0,0,52
    elif index<104: return 1,53,103
    elif index<145: return 2,104,144
    elif index<166: return 3,145,165
    elif index<197: return 4,166,196
    return -1

values = dict()
for i in range(len(sim_matrix)):
    row = sim_matrix[i]
    l, mini, maxi = getRange(i)
    tool = order[l]
    x = np.array([k for j,k in sorted(row.copy(), key=lambda y: y[0])
                  if mini<=j and j<=maxi and i!=j])
    if tool not in values: values[tool] = []
    else: values[tool].extend(x)

for key, val in values.items():
    print(key,':')
    print('mean =',np.mean(val)*100,'%')
```

```
    print('median =',np.median(val)*100,'%')
    print('std =',np.std(val)*100,'%','\n')
```

```
OKM :
mean = 13.335245823166384 %
median = 12.164767647910601 %
std = 7.742948860349682 %

RR :
mean = 7.207234551634804 %
median = 5.840786763530877 %
std = 7.201341960580623 %

CP :
mean = 8.960207043365546 %
median = 7.945288841586877 %
std = 6.699673111808958 %

L :
mean = 9.512416798968887 %
median = 6.640961257471025 %
std = 7.935171036945507 %

I :
mean = 12.012669313975373 %
median = 11.508751891801976 %
std = 8.024130745512133 %
```

```
def simi(i, graph,xx,name, logscale):
    x = xx.copy()
    x.sort()
    x.insert(0,0)
    y = list(range(len(x),0,-1))
    y = np.array(y) / ((groups.size()[order[i]]-1)**2)
    y[0] -=1
    x1,x2,y1,y2 = graph.axis()
    graph.axis((x1,0.4,y1,y2))
    graph.fill_between(x,y, step="pre", alpha=0.2)
    graph.plot(x, y,drawstyle="steps", alpha=0.9)

    if logscale: graph.xscale('log')
    graph.set_title(name)
    return x,y

def make(logscale=False):
    figs = [ax0, ax1, ax2, ax3, ax4]
    plt.tick_params(labelcolor="none", bottom=False, left=False)
    for i in range(len(figs)):
        x, y = simi(i, figs[i], values[order[i]], full_labels[i], logscale)
        ax5.fill_between(x,y, step="pre", alpha=0.2)
        ax5.plot(x, y,drawstyle="steps", alpha=0.9)
```

```
    ax5.legend(full_labels)
    #plt.xlabel("Similarity threshold [%]")
    #plt.ylabel("Percentage of papers")
    if logscale: ax5.xscale('log')
    ax5.set_title('overview')
    x1,x2,y1,y2 = ax5.axis()
    ax5.axis((0,0.4,0,1))

    fig.tight_layout()
    plt.show()
fig, ((ax0, ax1), (ax2, ax3), (ax4, ax5)) = plt.subplots(nrows=3, ncols=2, figsize=(15, 12))
make()
```



**UNIVERSITÉ CATHOLIQUE DE LOUVAIN**
**École polytechnique de Louvain**
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**