University of Helsinki
Department of Digital Humanities
Language Technology

Master's Thesis

Creating Maps of Science Using Topic Models; A Reproducibility Study

Yu An
014699180

Supervisors:
Dorota Głowacka
Alan Medlar

31.10.2020

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet – Faculty Faculty of Arts Department of Digital Humanities	Koulutusohjelma – Utbildningsprogram – Degree Programme Linguistic Diversity and Digital Humanities

Opintosuunta – Studieinriktning – Study Track Language Technology
Tekijä – Författare – Author Yu An
Työn nimi – Arbetets titel – Title Creating Maps of Science Using Topic Models; A Reproducibility Study

Työn laji – Arbetets art – Level Master's thesis	Aika – Datum – Month and year 11/2020	Sivumäärä– Sidoantal – Number of pages 52

Tiivistelmä – Referat – Abstract

Maps of science, or cartography of scientific fields, provide insights into the state of scientific knowledge. Analogous to geographical maps, maps of science present the fields as positions and show the paths connecting each other, which can serve as an intuitive illustration for the history of science or a hint to spot potential opportunities for collaboration. In this work, I investigate the reproducibility of a method to generate such maps. The idea of the method is to derive representations representations for the given scientific fields with topic models and then perform hierarchical clustering on these, which in the end yields a tree of scientific fields as the map. The result is found unreproducible, as my result obtained on the arXiv data set (~130k articles from arXiv Computer Science) shows an inconsistent structure from the one in the reference study. To investigate the cause of the inconsistency, I derive a second set of maps using the same method and an adjusted data set, which is constructed by re-sampling the arXiv data set to a more balanced distribution. The findings show the confounding factors in the data cannot account for the inconsistency; instead, it should be due to the stochastic nature of the unsupervised algorithm. I also improve the approach by using ensemble topic models to derive representations. It is found the method to derive maps of science can be reproducible when it uses an ensemble topic model fused from a sufficient number of base models.

Avainsanat – Nyckelord – Keywords
machine learning, text mining, topic models, scientometrics

Säilytyspaikka – Förvaringställe – Where deposited
Keskustakampuksen kirjasto

Muita tietoja – Övriga uppgifter – Additional information

Contents

1 Introduction

1.1 Motivation and Research Questions

Boundaries between scientific disciplines are not always clear-cut: different fields and sub-disciplines within the same field can rely on the same fundamental principles and discoveries; articles from two different fields can cite the same papers from a third field; a single study may involve researchers from different backgrounds. Such collaborations generally exist in academia and can be traced back through data. For example, citations in scholarly articles can be seen as a link from one article to another, and thus one field to another the articles respectively belong to. Given enough of such data, a graph of science can be built to illustrate the relationships among the disciplines. Scientometrics uses the term "maps of science" to refer to this kind of visualisation, which shows the structure of science in a straight-forward way.

Among the diverse approaches to map the science, Huang et al. (2019) proposed a full-text analysis method based on topic modeling. The idea was to represent disciplines with vectors, of which each entry correspond to the KL divergence (Kullback and Leibler, 1951) from the content of a given section to the full-text of this field's literature. The divergence value describes the information loss occurring when using the section as a summary of the full-text. Similar vectors, which imply similar subfields, are placed close to each other. The space can be visualized as a tree structure by performing hierarchical clustering on the vectors. The resultant map (Figure 1) showed an indicative pattern: for example the sub-clusters at the upmost node seem to correspond to theoretical and applied disciplines respectively.

This thesis aims to expand Huang et al. (2019)'s work by exploring how reproducible the method is in building consistent maps of science. In this context, reproducibility refers to how good the method is in overcoming the variation brought by a) different samples and b) the nature of the algorithm, "sample randomness" and "algorithm randomness" using the terms from Dolnicar and Leisch (2010). The goal is to answer the following questions:

- **Research Question 1** Does the method from Huang et al. (2019) produce consistent results?
- **Research Question 2** What are the main sources of variation in the algorithm and the data set?

- **Research Question 3** Using our findings from RQ1 and RQ2, what improvements can be made to the method?

RQ1 and RQ2 aim to evaluate the reproducibility of the method in front of different samples, that is, how good it is at overcoming "sample randomness" (Dolnicar and Leisch, 2010). In my case, the samples are distinct groups of scholarly articles that are independently collected and may vary in wording and writing styles. A robust, reproducible method should be able to see through the variations in the sample and capture the common characteristics of the field, thus should produce consistent science maps even from different data sets. To test that, I first explore RQ1 by replicating Huang et al. (2019)'s method. A group of new maps is derived from an updated, larger set of data, whose consistency would speak about the stability of the method in front of different samples. It would be impossible and futile to replicate the exact study as in Huang et al. (2019), for I have no access to the annotation they used, which makes it impossible to replicate the exact data as they used in the study. On the other hand, the larger data set should be a more representative sample of the population. The new data is constituted by the Computer Science papers from arXiv [1], a preprint repository, the submissions between July 1991 and July 2019. The data set is referred to as *130K data set* from now on. Huang et al. (2019) used much fewer papers, which takes only 24.1% of the literature on the server.

Continuing from RQ1, RQ2 seeks to track the source of variation, whether it is caused by the different structure of the data or the stochastic properties of the algorithm. A factor in the data that I find may affect the result is the distribution of the articles' categories: for each article in the data set, there is an attribute of category to indicate the sub-field(s) it belongs to; it is noticed that the count of papers across categories varied widely, and some categories are more correlated than others. To test if these factors are the source of variation, two experiments are conducted to contrast the results from using the data with and without the balanced counts respectively. The data set without imbalanced categories is constructed by artificially balancing the aforementioned categories in *130K data set*. The two sets of results, if concordant, exclude the possibility that the variation is due to the different data. Otherwise it should be due to the randomness in the algorithm.

The stochastic nature of algorithms can be caused by, for example, using a random start point during the optimization (Dolnicar and Leisch, 2010). Based on the findings from RQ1 and RQ2, I test the effect of some machine learning techniques to see if they are useful in improving the reproducibility.

[1]https://arxiv.org/

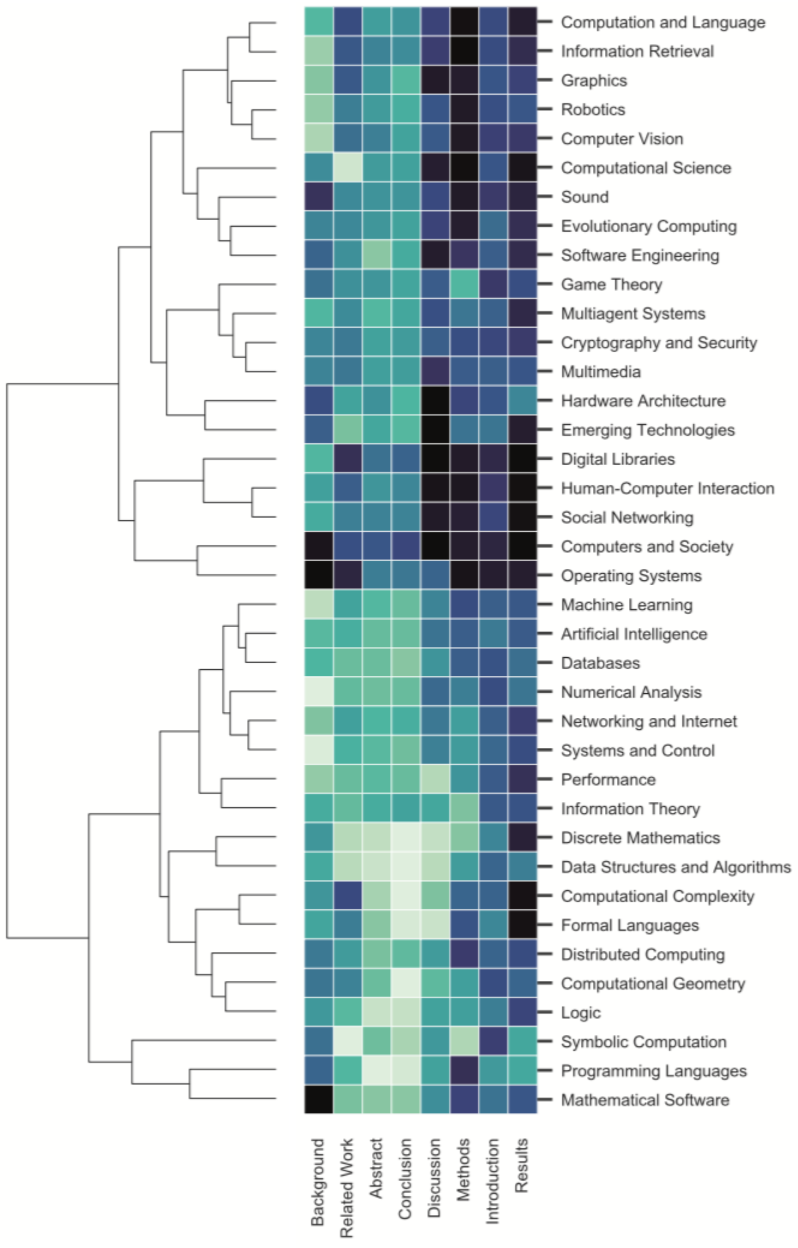


Figure 1: Map of Computer Science from Huang et al. (2019), based on 35,137 papers from arXiv submitted 2007-mid 2015.

1.2 Overview of the Work

The remainder of this work is organized as follows: Section 2 introduces the topic of maps of science, briefly reviews the approaches that have been developed for it and specifically explains the theories used in Huang et al. (2019)'s approach, LDA topic models (Paul and Girju, 2009). Section 3 describes the data used to reproduce the study, arXiv Bulk Files. Section 3 shows excerpts of the metadata and some descriptive statistics, which, as mentioned before, reveals the factors that could impact the reproducibility. Section 4 offers the theories behind the series of algorithms that compose the method of interest. Section 5 covers specifics of the experiments to approach the research problems, from data pre-processing to obtaining the maps. Lastly, Section 6 answers the research questions, exhibiting the maps derived in each step and the corresponding objective metrics to evaluate them, which add up to the conclusion in Section 7. Section 7 also reflects on limitations of this work and suggests on possible improvements.

2 Background

In the following, I introduce the topic of maps of science and summarise the approaches to creating the maps. Topic modeling, a crucial piece that composes the method to reproduce, is specifically reviewed, for it is potentially a factor that can affect the reproducibility of the method.

2.1 Maps of Science

As the name suggests, "maps of science" (Small, 1999) are drawings that present science areas in the form of maps. Such a map illustrates the relationships among scientific fields and puts the fields in an organized form, which is intuitive as human mind naturally gravitates to organized items (Nilson and Goodson, 2017). Seeing the connections between fields can provide insights into the state of scientific knowledge, which is "the first requirement of good history of science" (Small, 1999; Holton, 2000). A practical use of science maps can be, for example, for researchers to spot potential collaborators from outside of their own fields (Boyack, 2009). Nowadays maps also serve as part of the design for the guiding interface of digital libraries, which help users to navigate through plethora of resources (Börner et al., 2012; Börner, 2004).

The process of making science maps typically boils down to completing two tasks, "classification and visualization" (Boyack and Klavans, 2014). Classification involves dividing the disciplines into distinct groups, which serves as the input to the tools for visualization. Therefore a mapping approach is distinguished from the unique combination of the methods for the two tasks, while the major difficulty is usually on the task of classification (Boyack and Klavans, 2014). As a result, this review focuses on the diverse methods applied in the classification part.

As a review of the approaches to making maps of science, Suominen and Toivanen (2016) argued that the method for classification usually falls into one of the following categories: co-citation analysis, co-word analysis, co-author or co-affiliations analysis, and hybrid use of these. The first attempt at building maps of science dates back to 1974, when Griffith et al. (1974) mapped the most cited subfields of biomedicine, physics, and chemistry based on 1,310 references (Boyack and Klavans, 2014). In the work, articles were viewed as nodes in a graph and connected to one another by referencing. It is a typical case of co-citation analysis, as only the references in the data of papers were used and the content in the full-text was ignored. Co-author or -affiliations analysis (e.g. Peters and Van Raan (1991)) works in a similar way, focusing on the metadata of the articles. The way to analyse the body of the papers, i.e. the semantic content, are called co-word analysis (Callon et al., 1991).

Besides scientific publications, there are other resources to derive representations for the scientific fields. For example, Boyack and Klavans (2014) built a high-resolution map with "clickstream" (Boyack and Klavans, 2014) data from scholarly webs. A clickstream represents a series of clicks a user makes on the web portal to move from one journal to another. Combining it with user information (e.g. users' research interest etc.), Boyack and Klavans extracted a "journal network" from the clickstream, which reflected the collaboration among scientific domains. This new way to derive the relationships uses the data from outside of the content and cannot be categorised to any kind of analysis mentioned above.

According to Suominen and Toivanen (2016)'s definition, Huang et al. (2019)'s approach is a co-word analysis. The approach represents the disciplines by probabilities inferred from the content of full-text. The probabilistic model used in the inference, topic models, is introduced next.

2.2 Topic Modeling

Topic modeling is a statistical method typically used for the task of representing documents. It is in a sense similar to probabilistic language models (LM), for both kinds of models assign probabilities to a piece of text. The difference is that a language model would represent a document by assigning a joint probability of co-occurring n-grams (Srikanth and Srihari, 2002), whereas a topic model takes a document as mixture of a fixed number of topics and represent the document with the probability distribution of the topics. Each of the topics (see the left hand side of Figure 2) is a list of words that frequently occur together in the content.

In the context of statistical methods, a topic is a probability distribution over the topical words. Taking the first topic in Figure 2 as an example, it seems to express the topic of gene, and the numbers following the words indicate the conditional probability of the words.

Likewise, a document represented by the topic model is captured by a probability distribution over topics. As in the right hand side of Figure 2, the words in the document correspond to the topics detected in the corpus, which assigns topic distributions to the documents. Hence topic modeling holds the assumption that a document is composed of a fixed number of topics, and the representations of documents are probability distributions that describe the topic proportions. Formally, the task of topic models is to estimate the following distributions:

- Probabilities of word (w) in topics (t)

  $\phi_{wt} = p(w|t)$

- Probabilities of topics (t) in documents (d)

  $\theta_{td} = p(t|d)$

given the collection of texts n_{wd} as bag-of-words, that is, the count of word w in the document d.

Representing text as bag-of-words means topic models do not mind the order of the words in the text, and the generative process the models describe is in fact a process to sample words from some distribution. The distribution is determined by the assumption of the specific variant of topic models. Topic modeling in fact refers to a type of statistical methods that assume the topical structure in documents (Dalwadi, 2020). In other words, any algorithm that uses topics to represent the text can be called a topic modeling method.

In both the reference work and this work, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is used to model and represent the text data to prepare for generating maps of science. The formal task of LDA follows the general task of topic models that is stated above, while it holds the assumption that the document-topic (θ_{td}) and topic-word (ϕ_{wt}) distributions follows Dirichlet distributions.

The generative process can be summarised as a two-step procedure: first to choose a topic for the next word to write, then to pick a word from the list of words related to the topic. Pseudo-codes and illustrations of the generating process are detailed in the section for methodology (Section 4).

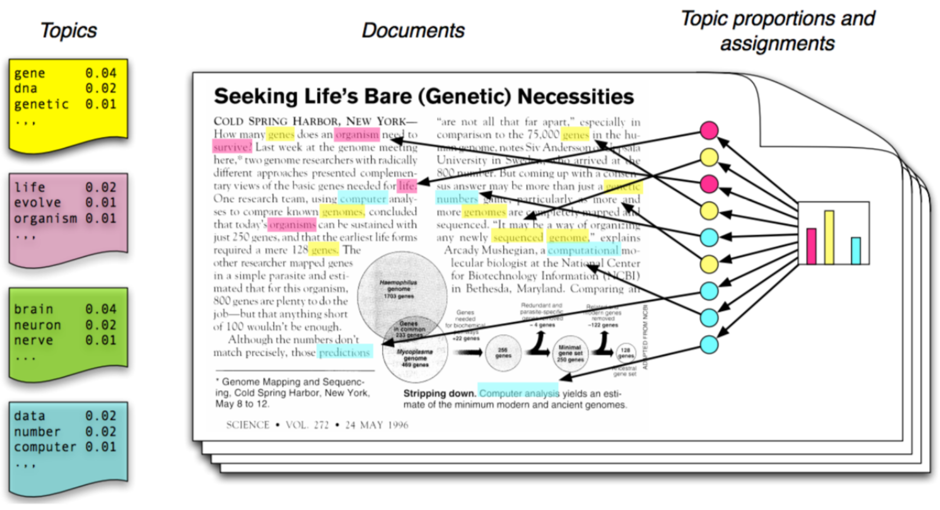


Figure 2: Topics in a document. An illustrative image from Blei (2012).

Topic modeling is in a sense a clustering algorithm as it aims to divide the words into similar groups and take the groups as topics. Hence "sample randomness" and "algorithm randomness", the factors in Dolnicar and Leisch (2010) to account for reproducibility of clustering algorithms, also applies to the research to it.

Topic modeling is commonly used in information retrieval applications such as search engines, for the topics are interpretable and users can utilise the topic to sort a document on relevance (Dalwadi, 2020). Suominen and Toivanen (2016) has used this approach to present a map of Finnish science and concluded the benefits and drawbacks of the result in contrast with those from human-reasoning.

Tang et al. (2014)'s research gives a series of caveats on the use of the model and the possible limiting factors. The empirical study reminds us of some prerequisites to use topic modeling for the data: whether the data contains a sufficient number of documents; a sufficient number of words in the documents etc. (Tang et al., 2014). The quality of fit between the data and method is examined and reported in Section 3.1.

3 Data

Data need to be collected and examined prior to performing the analyses. In this study, two data sets are used to reproduce the previous work, namely the *130k* and *60k* data set, which are introduced in the following.

This section explains how the data sets are collected and organized, and the findings from the exploratory analysis. Data science techniques, such as parsing and transforming the raw data, are applied to construct the data sets. The pre-processed data is directly usable for the analysis described in Section 5.

3.1 Data Source: arXiv

This work uses scholarly articles in arXiv. arXiv is a popular digital repository to deposit academic publications. By 2017, arXiv has hosted 23% of papers from the most selective conferences in Computer Science, along with numerous pre-prints, namely papers that have not been reviewed but have made public before the process (Sutton and Gong, 2017). Rich in both kinds of publications, arXiv is a good resource to acquire large quantities of text data.

A submission to arXiv is composed of metadata and the text. Authors submit the metadata as a form, where there are fields to fill and to specify the necessary information about the paper. In the case of Computer Science, the required fields are title, authors, abstract and categories (arXiv staff, 2004). The text submission contains the complete content of the paper, usually the source files and the rendered PDF, which arXiv requires to be one of the following formats: a) (La)TeX, b) PDF or c) HTML with JPEG/PNG/GIF images (arXiv staff, 2004). The majority format is TeX, the format that is recommended by arXiv (arXiv staff, 2004). Both parts are accessible via the service of arXiv Bulk Data Access. The metadata is returned as an XML file (see the excerpt in Listing 1), which has a structured format and allows to extract useful information about the articles.

The full-text access returns the source files of the articles. In this work, 145,428 papers are downloaded with the metadata through the access provided by arXiv. Those are all the papers submitted to Computer Science of arXiv between July 1991 and July 2019. Download made after this date may change, as authors can withdraw or edit their submissions. The data set is thus a snapshot of the database taken in July 2019.

```
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
    ="http://www.openarchives.org/OAI/2.0/␣http://www.
    openarchives.org/OAI/2.0/OAI-PMH.xsd">
<responseDate>2020-03-28T17:48:39Z</responseDate>
<request verb="GetRecord" identifier="oai:arXiv.org:0804.2273"
    metadataPrefix="oai_dc">http://export.arxiv.org/oai2</request
    >
<GetRecord>
<record>
<header>
<identifier>oai:arXiv.org:0804.2273</identifier>
<datestamp>2008-04-16</datestamp>
<setSpec>cs</setSpec>
</header>
<metadata>
<oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/
    oai_dc/" xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
    oai_dc/␣http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
<dc:title>
Object Re-Use & Exchange: A Resource-Centric Approach
</dc:title>
<dc:creator>Lagoze, Carl</dc:creator>
<dc:creator>Van de Sompel, Herbert</dc:creator>
<dc:creator>Nelson, Michael L.</dc:creator>
<dc:creator>Warner, Simeon</dc:creator>
<dc:creator>Sanderson, Robert</dc:creator>
<dc:creator>Johnston, Pete</dc:creator>
<dc:subject>Computer Science - Digital Libraries</dc:subject>
<dc:subject>
Computer Science - Networking and Internet Architecture
</dc:subject>
<dc:subject>C.2.3</dc:subject>
<dc:description>
The OAI Object Reuse and Exchange (OAI-ORE) framework recasts the
     repository-centric notion of digital object to a bounded
    aggregation of Web resources. In this manner, digital library
     content is more integrated with the Web architecture, and
    thereby more accessible to Web applications and clients. This
     generalized notion of an aggregation that is independent of
    repository containment conforms more closely with notions in
    eScience and eScholarship, where content is distributed
    across multiple services and databases. I provide a
    motivation for the OAI-ORE project, review previous
    interoperability efforts, describe draft ORE specifications
    and report on promising results from early experimentation
    that illustrate improved interoperability and reuse of
    digital objects.
</dc:description>
<dc:date>2008-04-14</dc:date>
<dc:type>text</dc:type>
<dc:identifier>http://arxiv.org/abs/0804.2273</dc:identifier>
</oai_dc:dc>
</metadata>
</record>
</GetRecord>
</OAI-PMH>
```

Listing 1: An example of metadata: the record of Article 0804.2273.

Not all of the papers go to the data set. As in the reference study, the content in the documents must be converted to word tokens, such that it can be further analysed. Documents that cannot be converted have to be abandoned.

Two types of such unusable documents are non-TeX articles and TeX articles that uses unsupported packages. Non-TeX articles, i.e. the submissions in PDF or HTML, take only a small fraction of the total, hence are ignored given the time and resource. The problem with unsupported packages is that these papers cannot be parsed by the tools I use for conversion (see Section 3.3.1), hence are excluded from the later experiments. As a result, the final data sets are composed only of those which are successfully converted by the pre-processing pipeline.

3.2 Data Organization and Confounding Factors

The data is organized in the structure outlined in Figure 3. The structure under the folders, the XML files, are the result from pre-processing the TeX articles.

The data sets used to reproduce the study are plain text parsed from the XML. The plain text composes two data sets, each of which covers a different number of articles. The *130k data set* includes all of the articles whose

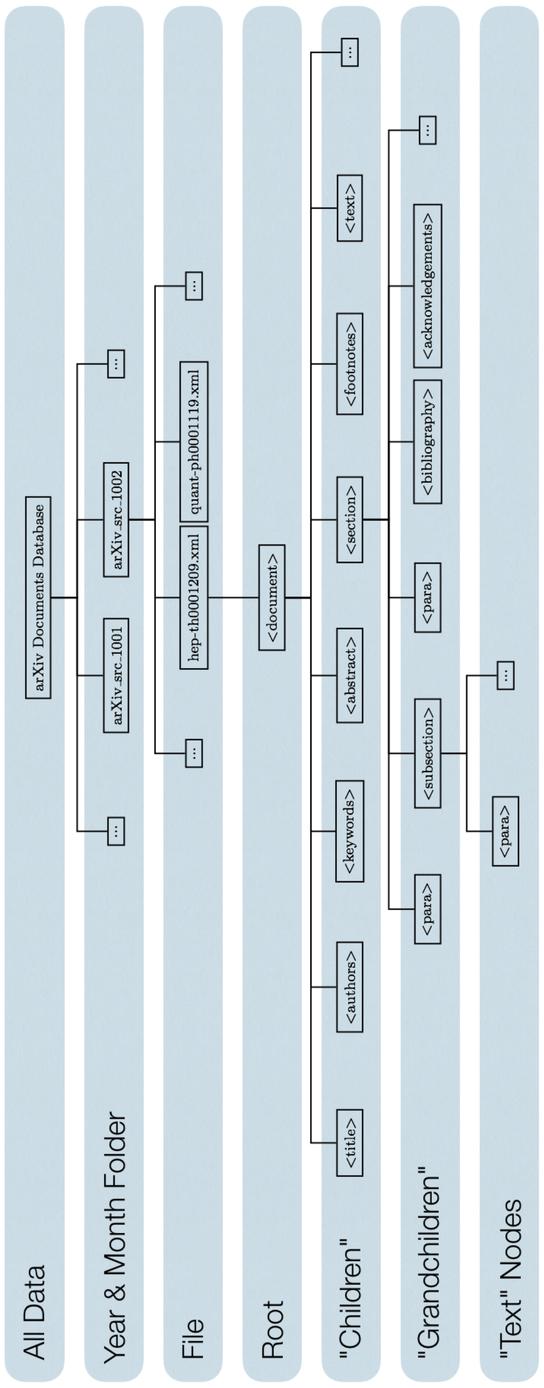


Figure 3: Data Organization Tree

XML are successfully parsed, that is 131,703 documents. The other data set includes papers sub-sampled the *130k data set*, which are 58,940 articles and is referred to as the *60k data set*.

Each article within the data sets is presented in two forms. One is a single document that contains the full-text of the article. The other version splits the article by sections, each of which is a document. The two versions serve the different stages in deriving representation vectors for the sub-field, which is covered in Section 5.2.

The 60k data set is used to contrast the results from using the 130k data set. Comparing the reproduced work from the 60k and the 130k data set demonstrates the effect of confounding factors in the data. The confounding factors involve the skewed distribution of categories and the correlating labels, which will be detailed next.

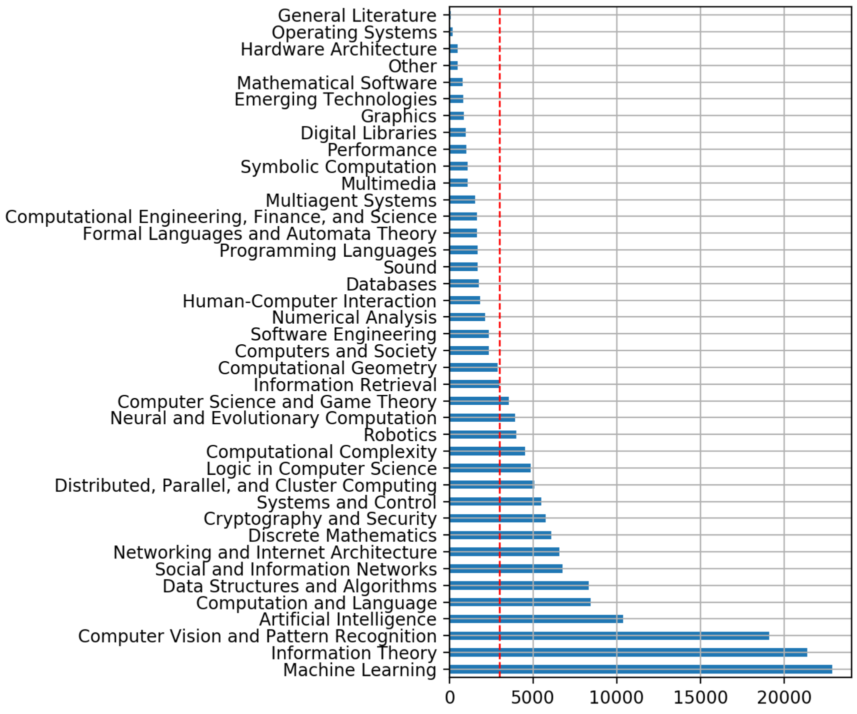
Listing 1 shows the pseudo-code for the sub-sampling procedure. The procedure builds the 60k data set by gradually removing random documents category by category, until there are no more than 3000 papers under all of the categories. The goal of the procedure is to get a sample with a category distribution that is as uniform as possible and is similar to the original. In this way, the result from the 60k data set can contrast with the 130k data set. Figure 4 illustrates the category distributions of the two data sets.

In Listing 1, documents are removed gradually to avoid down-sampling the categories that correlate to other fields. It is the other confounding factor that is found in the 130k data set. As an illustration, Figure 5 shows the distribution of the labels in Machine Learning. Clearly, the labels of Computer Vision and Artificial Intelligence are often associated with Machine Learning.

Both of the data sets should be suitable to be represented by LDA topic models according to Tang et al. (2014)'s caveats. The 131,703 and 58,940 documents with 5,313 and 4,653 respective average word counts satisfy the definition of "many" "long" documents.

3.3 Pre-processing

This section walks through the pre-processing procedure, where two tasks, data transformation and annotation, are completed consecutively. Being in TeX with markup tags, the raw data is not directly usable for the method to generate maps. To address the issue, I build a pre-processing pipeline to perform a series of transformations to remove the noisy characters without losing the word tokens. The clean data is then annotated, which provides all



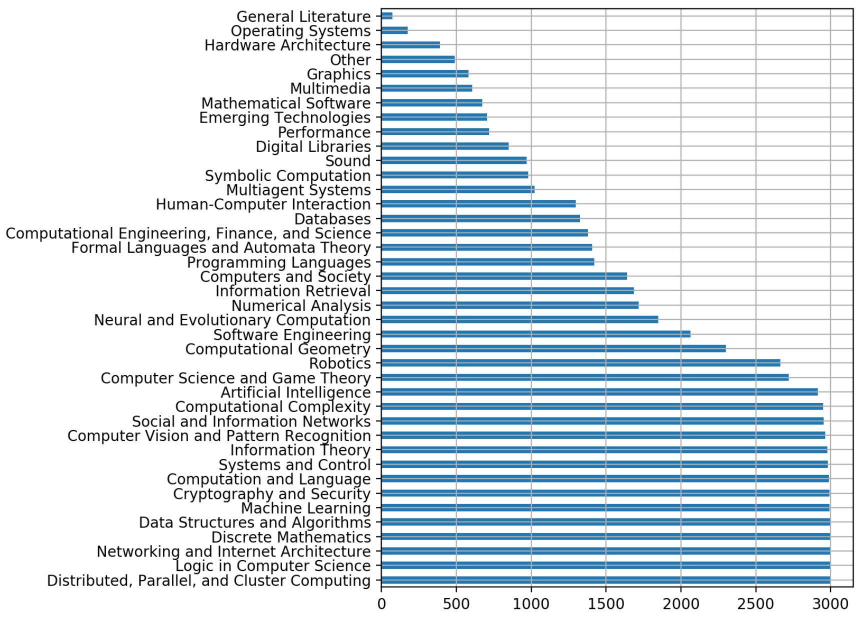


Figure 4: The distribution of category of 130k data set (up) and 60k data set (bottom). The sum of the frequencies does not equal the number of the papers, as a paper may belong to more than one categories. The dashed line shows the threshold of 3000.

Algorithm 1 Resampling method

```
1: procedure SUBSAMPLE(input $dataset$)
2:     Get category frequencies of the dataset
3:     $X \leftarrow$ the category of the most papers
4:     while Category $X$ that has more than 3000 papers do
5:         Randomly remove 100 papers under category $X$
6:         Update category frequencies of the dataset
7:         $X \leftarrow$ the category which has the most papers
8:     end while
9: end procedure
```

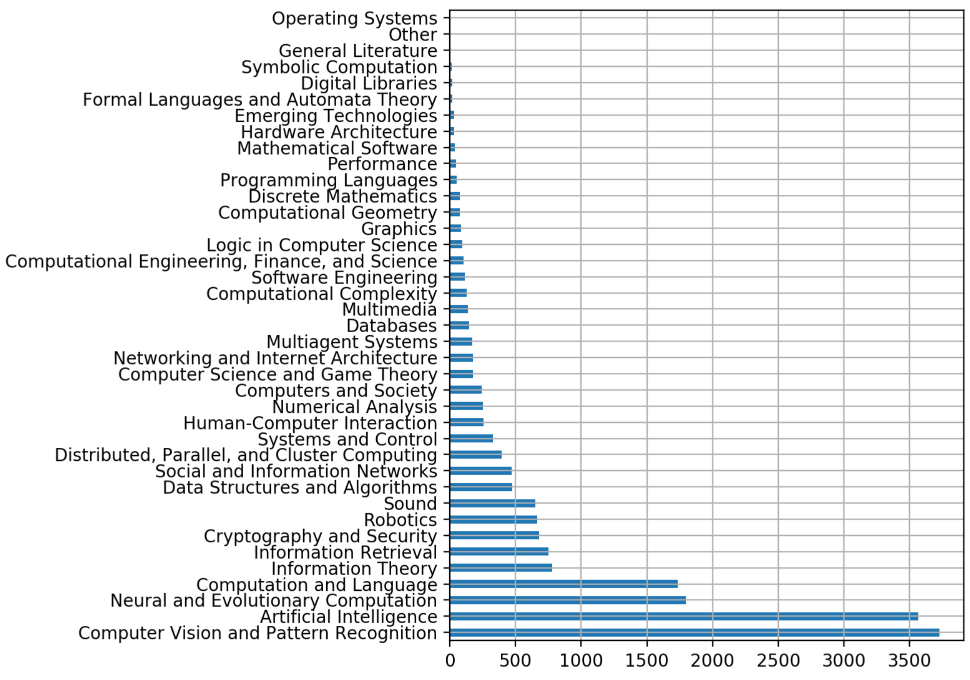


Figure 5: Category frequencies of the papers in Machine Learning. Here shows the most associated labels with Machine Learning, i.e. Computer Vision and Pattern Recognition and Artificial Intelligence. The fact that some fields tend to correlate with some other fields can be a confounding factor in the data.

the information needed to reproduce the method.

3.3.1 Data Transformation

From TeX to XML

The raw data from arXiv are (La)TeX source files. In these files, markup tags, maths, figures and tables mix with semantic content, which makes the data noisy and affect the performance of the topic models (Fan and Stewart, 2015).

Avoiding the noise requires removing the markup tags; however, this would also take the information about the structure of the articles. The information should be kept, for I need to extract individual sections for the later analysis. Therefore, the first step in pre-processing is to transform TeX to an intermediate format, a structured data format such as XML or JSON, such that information about the boundaries between sections can be kept before I remove the markup tags.

The tool I use for this task is *LaTeXML* (Miller, 2019), which transforms TeX into XML, a working format that allows to keep the structure information. *LaTeXML* 0.8.4 handles 90.6% of the 145,428 articles, in which the failed cases are empty files and are left out from the data set. The failures are caused by either the non-TeX format (i.e. PDF or HTML) of the source file or packages which are not supported by *LaTeXML* 0.8.4. Successful conversions yields readable XML, which means the files can be parsed by using the ElementTree[2] module.

From XML to "Clean" XML

The structure of the resultant XML is somehow still complicated. Markup tags, tables, figures and mathematics and other non-semantic content are just wrapped by the XML elements, not removed. An algorithm to extract content from the XML would require a lot of filtering, which is not handy. To smooth the extraction, I wrote another script to simplify the structure of the XML.

The algorithm used in the script represents XML as a tree and forces it to a pre-determined structure that is shallower and removes the formatting commands. The structure is illustrated in Figure 3.

[2]https://docs.python.org/3.5/library/xml.etree.elementtree.html

Under the root, the direct children are the following nodes: <title>, <abstract>, <authors>, <section>, <chapter>, <footnote>. The <section> and <chapter> nodes have children of <para>, the paragraph division. Apart from <section> and <chapter>, all the nodes have text directly wrapped within the node tags, which is parsed as the `.text` attribute of the nodes. Extracting a section means parsing the <section> or the <chapter> node, and extracting full-text means parsing all the nodes into `element` objects and join all the `.text` properties.

This step processes all the XML produced from the last step. No data item is lost.

From XML to Tokens

Another script (see the pseudo-code in Algorithm 2) is written to extract the word tokens from the XMLs. The format of the output documents is one-paragraph-per-line, plain text. As in the reference literature, punctuation, maths and numbers are removed; all the word tokens are lower-cased. This format makes the data directly usable for the later analysis.

As mentioned in Section 3.2, the extraction makes two versions for each of the articles to serve for model learning and inference respectively. The full-text version of the article is composed of the string in the `.text` attribute from all the nodes in the XML. The section version of the articles contains the string in `.text` attribute of the `<section>` or `<chapter>` nodes.

A separate text file records the filenames and the corresponding section headings. In both versions of articles, the abstract is extracted from the metadata instead of from the text submission. It is to ensure to extract the abstract for every article, in case the algorithm does not recognise it from the XML.

3.3.2 Annotating Sections

According to the method to reproduce, sections extracted from the articles are required to be categorised into a set of canonical classes. It is for computing the representation vectors in the later analysis, as each dimension of the vector corresponds to the average information loss from the article content to the content of the individual sections that are categorised into a canonical class. To know which canonical classes the sections account for, the sections must be categorised.

As in the reference study, the classification is done manually with a three-

Algorithm 2 XML File Parsing Algorithm

```
Load dictionary *heading2section*
Get path()
**for** each filename in compilation of files **do**
    Use ET.parse(path + filename) to establish tree
    Use tree.getroot() to establish root
    Create a master dictionary for the article; keys: "fulltext", "abstract", "introduction", "background", "related_work", "methods", "results", "discussion", "conclusion", "backmatter"; values: ""
    Use root.findall('.//*') to find all elements
    **for** each element in elements **do**
        Remove namespaces in the tag
    **end for**
    Use root.findall('./*') to find all direct nodes of root
    **for** each node in direct nodes **do**
        Use element.get('title','') to get the heading
        Use "".join(element.itertext()) to collect text
        Match and join all the tokens with regular expression r"*[a-zA-Z]+(?:[-'][a-zA-Z]+)*"
        Lower the case
        Assign the result to text
        Append text to "fulltext" in master dictionary
        **if** heading is found in dictionary *heading2section* **then**
            Append text to corresponding canonical section
        **end if**
        Write out fulltext to the fulltext folder; filename: document id
        $i \leftarrow 0$
        **for** each key in master dictionary **do**
            **if** key is not "backmatter" nor "fulltext" **then**
                Write out section to section folder; pathname: key; filename: arXiv identifier + "_" + $i$
                Record tuple (filename, canonical section) to the log
                $i \leftarrow i + 1$
            **end if**
        **end for**
    **end for**
**end for**
```

annotator setup. The annotators, two Computer Science academics and I, label the sections independently. The classification for each of the sections is decided by a majority vote.

In practice, annotators label the data through a web application (Figure 6). Each annotator opens the application locally, annotates the data, and submits the text file produced by the application as the result.

The task is done by choosing the options that can describe the heading of the given section. A section can belong to one, more than one, or none of the categories. The canonical classes include abstract, introduction, background, related work, methods, results, discussion, conclusion and back matter. The rubric (see Figure 6) lists typical headings that occur in academic publications and is shown under the options as a reference.

As stated before, the final category of the headings are decided based on a majority vote. If 2 or 3 annotators categorise the heading the same way, the categories are the labels for the heading. In the end, 1013 unique headings are labeled, which accounts for 46.62% of the all 22746 sections. Those which are uncategorised occur only once in the corpus, hence are excluded from further analysis. The agreement among annotators are almost perfect, as the Fleiss' kappa, as a metric for inter-rater agreement that has a scale of 0-1, reaches a very high 0.90.

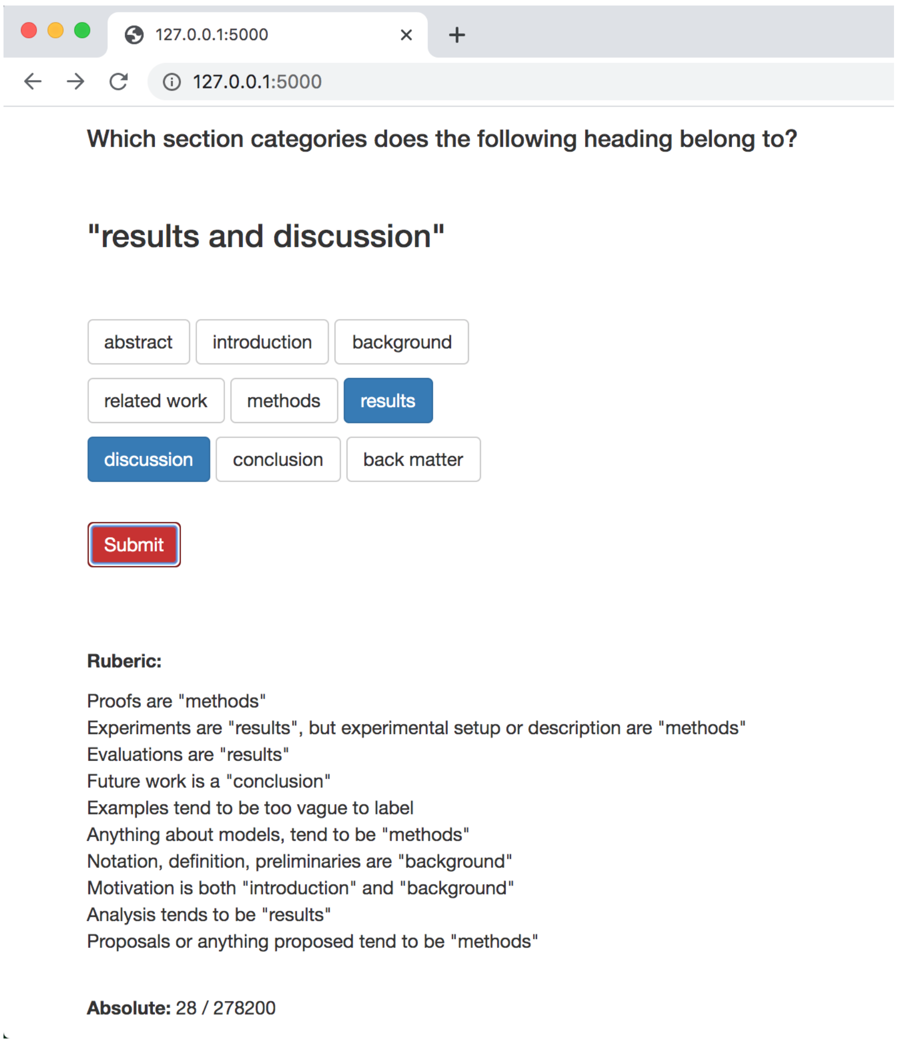
4 Methods

The original study (Huang et al., 2019) used a series of machine learning algorithms to generate a map of computer science. As a reproducibility study, this work repeats the workflow, gets a new set of maps for the same fields and compares the original and the new results.

This section explains the theories behind the series of algorithms and the metrics used in the comparison.

4.1 Latent Dirichlet Allocation (LDA)

Topic modeling, the statistical method used in the approach to reproduce, defines a process to generate documents. It is referred to as Latent Dirichlet Allocation (LDA). In the process, each document (θ) is considered as a distribution over a set of topics, and each topic (ϕ) a distribution over words. Instead of taking on a fixed value, these parameters have Dirichlet distri-



Which section categories does the following heading belong to?

"results and discussion"

Ruberic:

Proofs are "methods"
Experiments are "results", but experimental setup or description are "methods"
Evaluations are "results"
Future work is a "conclusion"
Examples tend to be too vague to label
Anything about models, tend to be "methods"
Notation, definition, preliminaries are "background"
Motivation is both "introduction" and "background"
Analysis tends to be "results"
Proposals or anything proposed tend to be "methods"

Absolute: 28 / 278200

Figure 6: An example annotator screen

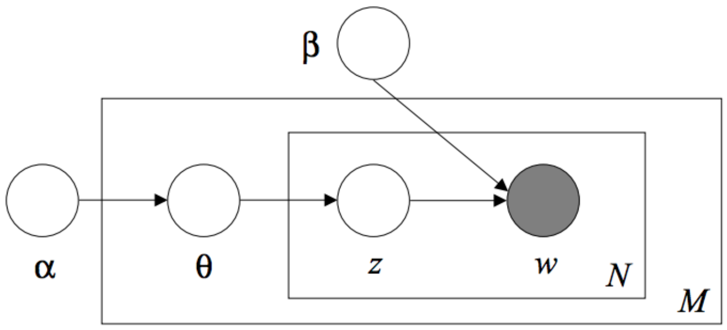


Figure 7: Graphical model of LDA from Blei et al. (2003). Circles represent the random variables denoted by the adjacent letter . The plates represent repeated generating process. The process indicated by the outer plate is repeated for M documents in the corpus, and the inner plate for each word in a document. The letters denoting random variables are consistent with the description in Section 4.1.

butions, which is the assumption of LDA as a Bayesian method. Heinrich (2005) summarises the generative process by the following steps:

- Define topics ϕ that account for the corpus, each of which is a probability distribution sampled from a Dirichlet distribution with parameter β ;
- Decide on the topic composition for the document, which should be sampled from a Dirichlet distribution ($\theta_d \sim Dir(\alpha)$);
- Decide on each word (w_n) by
  - Sampling a topic z from $Multinomial(\theta)$;
  - Sampling a word $w_{d,n}$ from $Multinomial(\phi_{z_{d,n}})$;

Blei et al. (2003) illustrate the generative process with a graphical model (Figure 7). Heinrich (2005) summarises the process with pseudo-code; see Algorithm 3. The output of the process would be a corpus, or a collection of documents. Formally, these are a series of posterior probabilities P$(w_n|\beta)$. The learning process of the model would be a different one where parameters for the needed distributions are estimated.

Algorithm 3 The Generating Process of LDA (Heinrich, 2005)

```
1: procedure (input: Number of topics (K), number of documents (D),
   number of words in each document (N), parameter for document-topic
   distribution ($\alpha$), parameter for topic-word distribution ($\beta$))
2:     for $k$ in $[1, K]$ do                          ▷ Initialise topics
3:         $\phi_k \sim Dir(\beta)$
4:     end for
5:     for $d$ in $[1, D]$ do                          ▷ Generate Documents
6:         $\theta_d \sim Dir(\alpha)$
7:         for $n$ in $[1, N]$ do                      ▷ Sample words
8:             $z_{d,n} \sim Mult(\theta)$
9:             $w_{d,n} \sim Mult(\phi_{z_{d,n}})$
10:        end for
11:    end for
12: end procedure
```

In practice, learning a LDA model requires estimating two parameters: α, which is the parameter of the Dirichlet distribution for the document-topic probabilities, and β, the parameter of the Dirichlet distribution for the topic-word probabilities. In this work, the learning process is implemented by Gensim (Řehůřek and Sojka, 2010). It fits LDA models with online variational Bayes (VB), which is one of the approximate algorithms to estimate the posterior distribution. The inference algorithm is different from Gibbs sampling, which is used in Huang et al. (2019) by the software MALLET (McCallum, 2002). Given that the inference, i.e. computing the exact posterior probabilities for the parameters, is intractable (Blei et al., 2003), the approximate algorithm may make a difference in the actual learning for topic models. I choose Gensim because it enables to train the LDA models with asymmetric prior Dirichlet, which option is not provided by MALLET. The difference in the inferring algorithm can be a factor in the stochastic nature of algorithm.

Section 5 details the settings which decide how the parameters are learned.

4.2 Kullback-Leibler Divergence

In this work, the representation vector for a discipline is composed of the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) values between the topic distribution of each section and that of the full-text. KL divergence is a metric that compares two distributions and expresses the difference with a scalar. In our case, it is used to compare the topic distribution of the full-text of a field with the topic distribution of the section from the field. Formally, it is expressed as

$$D_{KL}(P||Q) = \sum_i P(i) \log_2 \frac{P(i)}{Q(i)}$$

where Q and P are two probability distributions, i the covariates of the distributions. In our case, P would be the topic composition of full-text documents and Q of the section of interest.

A greater value in KL divergence indicates higher information loss. KL divergence is not symmetric, which means the KL divergence from Q to P is different from the KL divergence from P to Q. In my case, the topic distribution of a given section is plugged in to the place of Q and the distribution of full-text to P, not the other way around. The divergence describes the representativeness of the given part for the full-texts. The base of logarithm is set to 2, such that the divergence value can be interpreted in bits.

As an example, in the case of plugging in the topic distribution of abstracts and full-text in Operational Systems, the yielded divergence values answer the following question: For the scholarly articles in Operational Systems, how many bits of information is lost when the abstracts is used to represent the full-text?

4.3 Hierarchical Clustering

Hierarchical clustering is the analysis performed on the representation vectors in Huang et al. (2019) to create a tree typology as the map of science, which is also performed in this work.

The mechanism of hierarchical clustering is to first group pairs of similar data items, then take each of the groups as a unit and group the groups of pairs. This procedure repeats until all the groups are fused to one, i.e. there is no pair to fuse. The algorithm is called agglomerative hierarchical clustering among other variations (Nielsen, 2016). The shape of the final cluster depends on the settings that are specified by the user. Once the settings are determined, the result would be deterministic, which means the shape of the cluster is always the same on the same data.

Concrete considerations in using hierarchical clustering involve a) defining a dissimilarity measure for inter-observations and b) a dissimilarity measure for clusters. It is the part where there is no absolutely right or wrong answers;

decisions are made based on characteristics of data and the expectation for the end result (James et al., 2013). Here, I follow the settings used in Huang et al. (2019) as a reproducibility study.

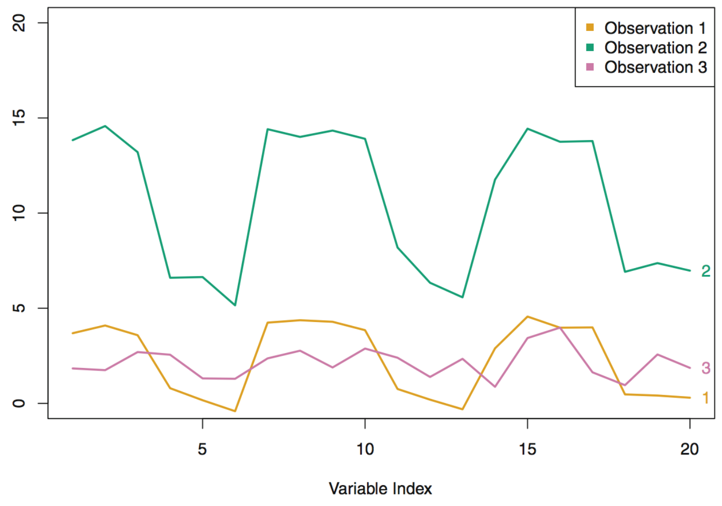


Figure 8: Graph from James et al. (2013) for Illustrative data with 20 features for each item. Observation 1 and 3 have a small Euclidean distance and a large correlation-based distance, while 1 and 2 have a small correlation-based distance and relatively larger Euclidean distance. For these observations, the choice of distance measure would greatly affect the result.

As mentioned above, dissimilarity measures are decided based on the kind of observations that are expected to go into the same subgroup. Alternatives for this parameter in practice are limited to two options: Euclidean distance or correlation-based distance. The difference between the measures is that Euclidean distance tends to classify vectors with similar magnitudes, whereas correlation-based measures prefer to group together vectors that takes a similar shape in profiles.

James et al. (2013) illustrated the point with a straight-forward example (Figure 8). In clusterings that use Euclidean distance, observation 1 and 3 would be grouped together due to the closer Euclidean distance by each index, whereas correlation-based measures prefer to combine 1 and 2 in spite of the greater distance. On the other hand, measures for each pair of clusters, or the type of linkage as the term, affects the fusions that happen at greater heights. James et al. (2013) also summarised the alternatives of linkage methods: Centroid linkage can bring "inversions", where clusters are merged at a lower height than the involved clusters thus is difficult to visualise; Single linkage often leads to skewed hierarchy; Complete linkage tends to generate balanced hierarchies; Average linkage is less conservative than complete, but the resultant cluster would be more balanced than the one from single linkage under the same setting.

In Huang et al. (2019), *Euclidean* distance is used as the dissimilarity measure between two data items and *complete* as the type of linkage. The choice indicates the magnitude of the KL divergences in the vectors weigh more than the correlations between the sections, and the groups of scholarly fields should be joined only when every field in the pair of groups is close enough to each other.

4.4 Tree Metrics

The conclusions in this work are based on quantitative comparisons between the resultant trees.

Two metrics are used for the comparison. One is a dissimilarity metric, Robinson-Foulds (RF) distance (Robinson and Foulds, 1981). The other is branch support (Bremer, 1994), a similarity measure. The metrics combined gives a complete picture about the consistency between the results.

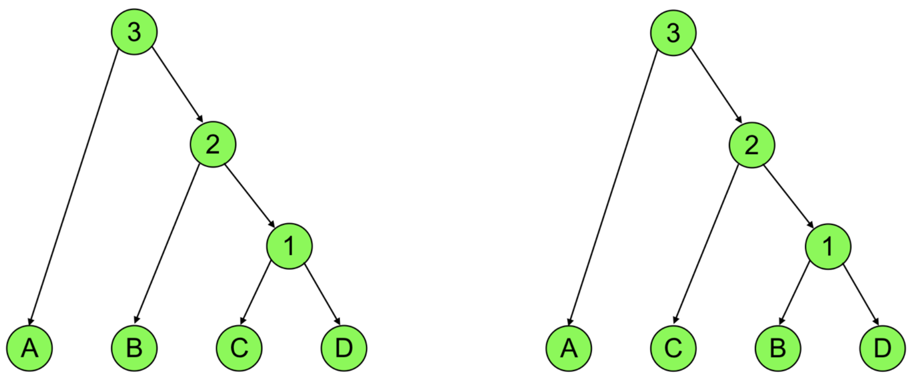


Figure 9: Example trees to illustrate how to calculate the RF distances between a pair of trees. The tree on the left is referred to as N_1 and the one on the right N_2. Letters denote leaves and numbers denote bi-partitions. The structure of N_2 is in fact the same with N_1 except Node B and Node C exchanges the places.

4.4.1 Robinson-Foulds Distance

Robinson-Foulds (RF) distance (Robinson and Foulds, 1981) is a metric of a single scalar that measures the dissimilarity between two trees. The intuition is to count the number of bi-partitions that are unique in one of the trees. Figure 9 shows two illustrative trees, which are referred to as N_1 and N_2 respectively. A non-terminal node (e.g. the nodes denoted by numbers in N_1 and N_2) splits the leaves (i.e. terminal nodes, denoted by letters in Figure 9) into two subsets, hence correspond to a bipartition. A bipartition is a set, or more specifically, a set of two sets, in which the elements are leaves in the tree. Table 1 shows the elements of the sets that define the bipartitions in N_1 and N_2.

1	{C}, {D}	1	{B}, {D}
2	{B}, {C, D}	2	{C}, {B, D}
3	{A}, {B, {C, D}}	3	{A}, {C, {B, D}}}

Table 1: The sets of leaves and the corresponding dividing bipartitions in the trees shown in Figure 9.

Let $i(T)$ denote the number of bipartitions in T and $e(T_1, T_2)$ the number of identical bipartitions in T_1 and T_2. Formally, the normalised version of RF distance is

$$RF(T_1, T_2) = \frac{i(T_1) + i(T_2) - 2e(T_1, T_2)}{i(T_1) + i(T_2)}$$

In the case of T_1 and T_2, the normalised RF distance between them would be 1, for there is no identical bipartitions between the trees. The normalised version counts the percentage of the unshared clusters, the range of it would be from 0 to 1.

It is worth noting that RF distance is sensitive to the difference in the groupings. High RF distance can occur between similar trees. As can be seen, T_2 is the result of swapping two random leaf nodes of T_1, but the RF distance between T_1 and T_2 rises to 1 from $RF(T_1, T_1) = 0$. It would be a more complete analysis if the reproduced structure can also be reflected. For this purpose, I also use a similarity measure, branch support, to compare the results.

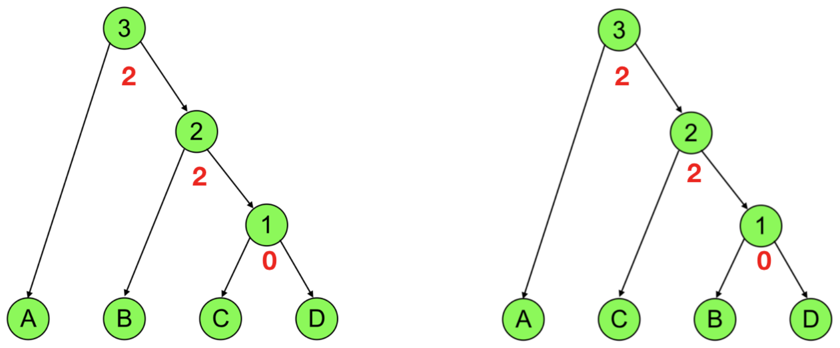


Figure 10: An illustrative graph for the branch support metric. The values next to the nodes indicate the branch support value of the structure the node represents.

4.4.2 Branch Support

Here I lend branch support, a concept from phylogenetics, to quantify the similarity between the trees.

In short, branch support counts the times that a branch appears in a group of trees. In biological researches, it is used for parsimony analysis, where multiple phylogenetic trees are constructed and compared by labelling how many times the branches appear in the reconstructed trees (Aluru, 2005). For example, "bootstrap percentage" is such a support metric, which is used to indicate the repeatability of branches in the trees constructed through bootstrapping the branches in a given set of trees. In practice, the metric is usually presented as numbers by the nodes on a graph that shows a tree topology, as it shows how many times the structure defined by the nodes is found in the reconstructed tree.

Figure 10 illustrates the metric using the same example trees as in the illustration for RF distances. The only structure that is not shared is the set {C,D}; otherwise, the clusters within clusters are considered as a set instead of a subset and thus do not affect the branch support of the parent nodes. Taking Node 2 as an example, the branch support of it is 2, as the structure {B,C,D} is found 2 times in the set of the trees. The metric captures the similarity of the sub-structures in the trees, which enable us to examine the results in a qualitative way.

This metric serves our purpose to spot reproduced sub-trees in the resultant maps, which qualitatively shows the extent to which the method is robust.

5 Workflow

This section explains how I derive a new group of maps of science and evaluate the consistency as the assessment to the reproducibility of the method. Analyses are conducted step by step, in the same order of how the subsections in this part are arranged.

5.1 Tuning and Training LDA Models

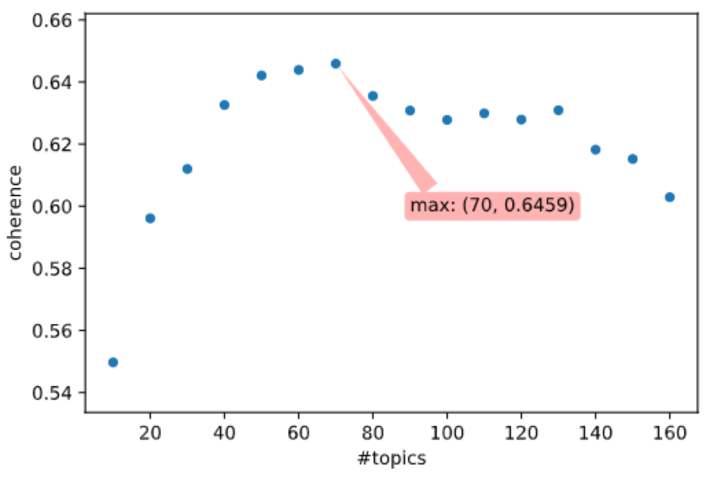
The method in Huang et al. (2019) maps a scientific field through vectorised documents. The first stage is to derive vectors to represent the fields, which requires training topic models. Before training, a list of settings must be known to yield working models. Searching for the best setting is the process of *tuning* the models, and involves multiple test runs. In this section, I focus on how the models are tuned, and the *training* of models is merely to put the found best setting to use.

I use the Gensim Python library's *ldamodel* implementation (Řehůřek and Sojka, 2010) to train topic models on the *130K* and *60K* data set respectively. The implementation uses online variational Bayes algorithm, which approximates the posterior probabilities of documents by passing "chunks" of documents through iterations, making the convergence faster for large data sets (Hoffman et al., 2010). To create models that can best represent the data, I:

- Pre-process the data;
- Search the optimal number of topics for the specific data set;
- Train LDA models.

The first step, a pre-processing procedure loads the documents into the format Gensim requires. Each document is taken as a single string and is tokenized through splitting by the occurrences of the white-spaces. Tokens are then filtered based on the frequency: Words that occur in less than 20 documents or more than 50% of the documents are removed.

For LDA models, the number of topics is a hyper-parameter that needs to be determined before training. The best setting can be found through test runs where a grid of candidate values are experimented. In the test runs for models that best represent the 130k data set, I create 16 different models by varying



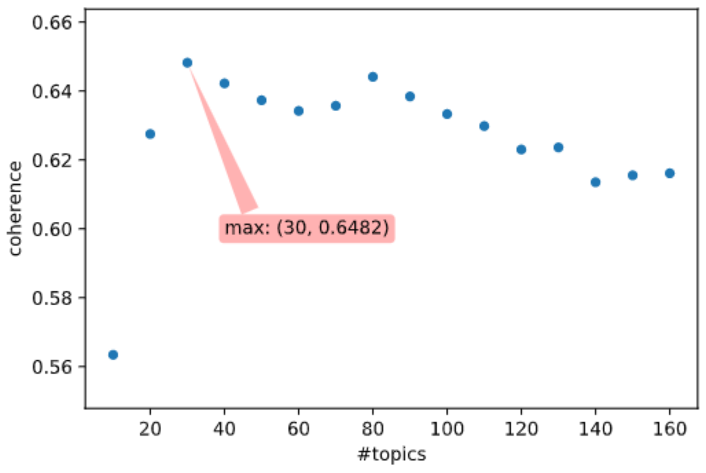


Figure 11: Topic coherence scores of the models from the test runs on the two data sets. Up: the 60k data set, bottom: the 130k data set.

the parameter from 10 to 160 in the increment of 10, which produces a series of topic coherence scores as the evaluation (Figure 11). The optimal setting should be the one that yields the highest coherence. The same searching procedure is repeated on the 60k data set.

Table 2 lists the settings I found best for training the final models as the representation for the 130k data set. The parameters that are not specified by the table follow the library default. Training models on the 60k data set uses the same settings, except the `num_topics` is set to 70. Figure 11 shows the coherence scores under different `num_of_topic` settings.

There are no rigorous studies on how to choose parameters for LDA models

Parameter	Description	Value
`chunksize`	Number of documents to be used in each training chunk	2000
`alpha`	A-priori belief for each topic's probability; The string 'auto' learns an asymmetric prior from the corpus	'auto'
`eta`	A-priori belief on word probability; The string 'auto' learns the asymmetric prior from the data	'auto'
`iterations`	Maximum number of iterations through the corpus when inferring the topic distribution	50
`num_topics`	Number of topics	30
`passes`	Number of passes through the corpus	10
`dtype`	Data-type to use during calculations inside model	np.float64

Table 2: Settings for the hyperparameters used in training the LDA models for the *130K* data set; parameters not mentioned in this table are set to library default. Settings for the *60k* data set are all the same, except that `num_topics` should be 70.

(Wallach et al., 2009). Except `num_topics`, our settings are generally designed in reference to the library default and Hoffman et al. (2010), where LDA was used to model a similar sized document set, a corpus of 3.3M articles from Wikipedia. The mini-batch size (`chunksize`) follows the library default, 2000, as Hoffman et al. (2010) finds that any size from 256 to 16384 would be proper. The Dirichlet priors, `alpha` and `eta`, are set to be asymmetric because symmetric priors is a special case of the asymmetric (Wallach et al., 2009). Such a setting enables the model to learn the priors from the data. The maximum iterations and the number of passes are tuned to 10 times of the value in the default to ensure convergence. For large corpora in my case, 64-bit data-type has to be used to avoid overflow.

In the end, 2 sets of 100 models are created to represent the fields that are reflected by the articles in the two data sets. Each set of the models are replicates yielded from repeatedly running the respective optimal setting 100 times on the used data set.

5.2 Obtaining Representation Vectors

A trained model can tell the topic compositions of full-text documents it saw in the training phase. To build representation vectors for scientific fields, the distributions of the distinct sections is still needed to compute the KL divergence. The section composition is obtained through inference, that is, the predicted distribution from the model given the section text.

Taking the full-text and section compositions as the input, Algorithm 4 fills an 8×38 table to show the representation vectors as a matrix, in which the rows correspond to the disciplines and columns the sections.

Algorithm 4 Computing Representation Vectors

```
1: procedure (input $fulltext\_composition$, $section\_composition$)
2:     for each field in 38 fields do
3:         $p\_i \leftarrow$ topic distributions for all full-texts in this field
4:         for each section in 8 canonical sections do
5:             $q\_i \leftarrow$ topic distributions for the section in this field
6:             Align $p\_i$ and $q\_i$ by document ids
7:             Get KL divergences for documents through element-wise multiplication between $p\_i$ and $log_2(p\_i)$-$log_2(q\_i)$
8:             Get KL divergence for (field, section) by averaging the KL divergences obtained from the last step by the number of documents
9:         end for
10:    end for
11: end procedure
```

5.3 Hierarchical Clustering Setup

The map of science is formed through visualising the representation vectors. As in Huang et al. (2019), hierarchical clustering is used as the visualising method.

I used *seaborn* (Waskom et al., 2014) Python package to perform the clustering. Each set of vectors, which consisted of 38 subjects with 8 sections for each subject, corresponds to one replication of the model training. The clustering algorithm divides the represented subjects into clusters and clusters of clusters, where the hierarchy is presented in the form of a tree. The result is illustrated by a heatmap and a tree.

The heatmap does not show the relationship between the fields; Rather, it indicates the relative magnitude of the KL divergences that compose the vectors. The algorithmic-specific variables follow the setting in Huang et al. (2019), that is, Euclidean distance with complete linkage.

5.4 Evaluating Consistency

Answering the research questions requires comparing the resultant maps. The comparison is demonstrated through the pairwise distances of the trees in the maps, or more specifically, the distribution of the distances between the results from multiple runs of the method. To understand the sample at hand, descriptive statistics for the distribution are informative enough.

To answer Question (1), which asks how reproducible the work in Huang et al. (2019) is, a distance distribution is obtained through one-versus-all comparison. The result from Huang et al. (2019) is compared with the two sets of trees from this work, which are induced by modeling the 130k data set and the 60k data set respectively. Since there are only 100 distances for each data set in this work, a histogram is enough to illustrate the distribution. If the method is reproducible, the distribution should show that the distances are mostly low, that is, it should cluster at 0 without spreading wildly.

The distances are to show how the method failed to reproduce. On the other hand, graphs showing the branch support measurement should discover the successfully reproduced local structures which the distance cannot reflect.

Question (2) is answered in a similar way to Question (1), that is by showing the distribution of RF distances between the trees. However, this time the tree pairs are from within the data sets rather than from across the two data sets.

Given two trees T_1 and T_2 from the same data set, i.e. either the 130k data set or the 60k data set, a distribution of the relative RF distance d is computed. Using the 100 trees from a single data set, I get $\binom{100}{2} = 4950$ dependent replications, which are the distances between all possible pairs of trees:

$$d_1 = RF(T_1, T_2), ..., d_{100} = RF(T_1, T_{100}), ..., d_{4950} = RF(T_{99}, T_{100})$$

Note the computation does not yield $100/2 = 50$ distances, which would be the result of sampling pairs of trees without replacement. Although the trees in the data set can be regarded as independent random variables in the function space, where independence is respect to the runs of modeling (Dolnicar and Leisch, 2010), the distance values are not independent and no statistical inference is conducted. The goal here is to understand the models at hand.

For Question (3), which asks if averaging the models reduces the variance in the result, one-versus-one comparison is conducted using the models from within the 130k data set. Before the comparison, pairs of ensemble trees are created by fusing N of the 100 models from the data set. Then the RF distances between the ensemble pairs are compared.

How the distances change as N increases indicates if the ensemble method is effective in stablising the result of topic models. More specifically, the procedure involves the following:

- Sample N of the 100 trees without replacement;
- Average the section vectors; renormalise them and recreate the tree,
- Repeat to get another ensemble tree;
- Calculate distance between 2 trees;
- Repeat the steps above 1000 times,

where N increases from 1 to 50. Finally, a heatmap is used to present the frequencies of distances between ensemble topic models, and a density plot to show the mean distance.

The experiments introduced above can be summarised to calculating RF distances and branch support values for tree pairs or trees from some kind of sampling.

A necessary step in this procedure is to translate the data structure into Newick form, which is the only format allowed for the tooling I use to compute RF distances (Arvestad, 2010), and branch support (BioPython package by Cock et al. (2009)). A simple Python script is developed to make the transformation; see the source code in Appendix A.

6 Results and Discussion

This section answers the research questions by illustrating and discussing the corresponding results. In addition, I present a map of computer science from the most stable model in the study along with some patterns I find from the map.

6.1 Reproducing the Published Results (RQ1)

The tree produced by Huang et al. (2019) is not reproducible on my expanded data set.

Figure 12 shows the result of the one-versus-all comparison between the tree in Huang et al. (2019) and its replicates built in the present work. Each distribution in the histogram is composed of 100 observations, which are distance values computed by comparing Huang et al. (2019)'s work with the 100 trees from the 130K data set and the 60k data set respectively. The phylogram shows the branch support values for the branches in Huang et al. (2019), which is obtained by comparing it to the trees for the 130k data set. All the nodes with no values specified are of 0 support.

Both the measurements indicate the model by Huang et al. is unreproducible. First consider histogram in Figure 12. It is clear that all the distance values between Huang et al. (2019)'s tree and the newly built ones, regardless of the data set from which they are inferred, are greater than 0.94. Both of the distributions peak at the highest distance value, 1.00, which comprises about 50% of the cases, indicating a lot of the reproduced works do not share any structure with the original. This aligns with the result from the branch support measurement, where most of the branches have 0 support, meaning no identical structures are found in the trees for the 130k data set.

The inconsistency mostly occurs in the early splits, namely the lowest branch nodes. As a consequence, the ancestor nodes of the early splits have 0 support. The only branch that is somehow reproduced is (ET, CC) (Emerging Technology, Computational Complexity), for which the branch support is 29, meaning that 29/100 trees featured the same branch. Considering most of the structure is of 0 support, I conclude the tree produced by Huang et al. (2019) is not reproduced.

The fact that the two distributions are similar proves the bias in the data do not have an obvious effect on the stability of the method. Compared to the 130k data set, the texts in 60k data set has a more balanced distribution of categories and fewer number of papers. Given that the performance of the mapping method almost ignores the difference in the data, it could be that the algorithm causes the inconsistent result.

6.2 Consistency of the New Results (RQ2)

The results are inconsistent even when they are induced from the same data. Figure 13 shows the result in the two measurements from the all-versus-all comparison between the trees built in this work, which are induced from models trained on the same data set. The distribution of RF distances is rather scattered, hence kernel density estimates instead of a histogram is used to illustrate the distribution. Only one of the results from the branch

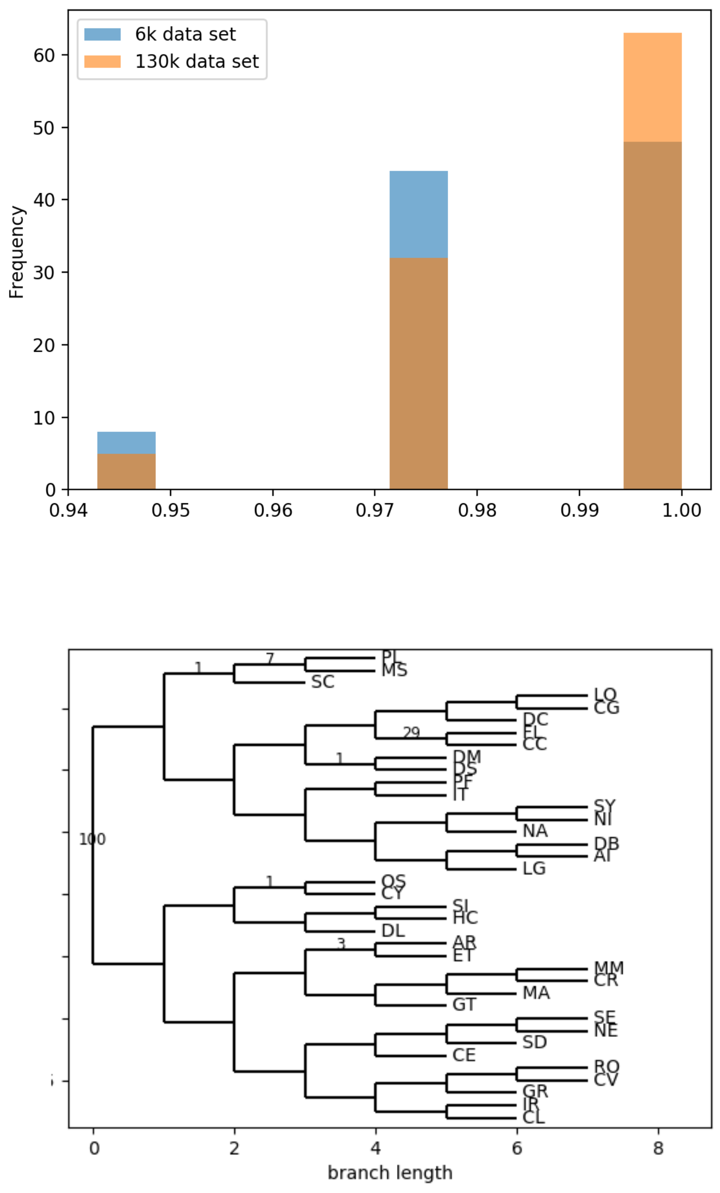


Figure 12: Top: Histogram of RF distances between Huang et al. (2019)'s work and the trees derived from the 130k and 60k data set respectively. The bins for the two data sets overlap, hence the third color. Bottom: Branch support between Huang et al. (2019)'s work and trees from 130k data set, subfields specified using the acronyms in arXiv. See the corresponding full names in Table 3.

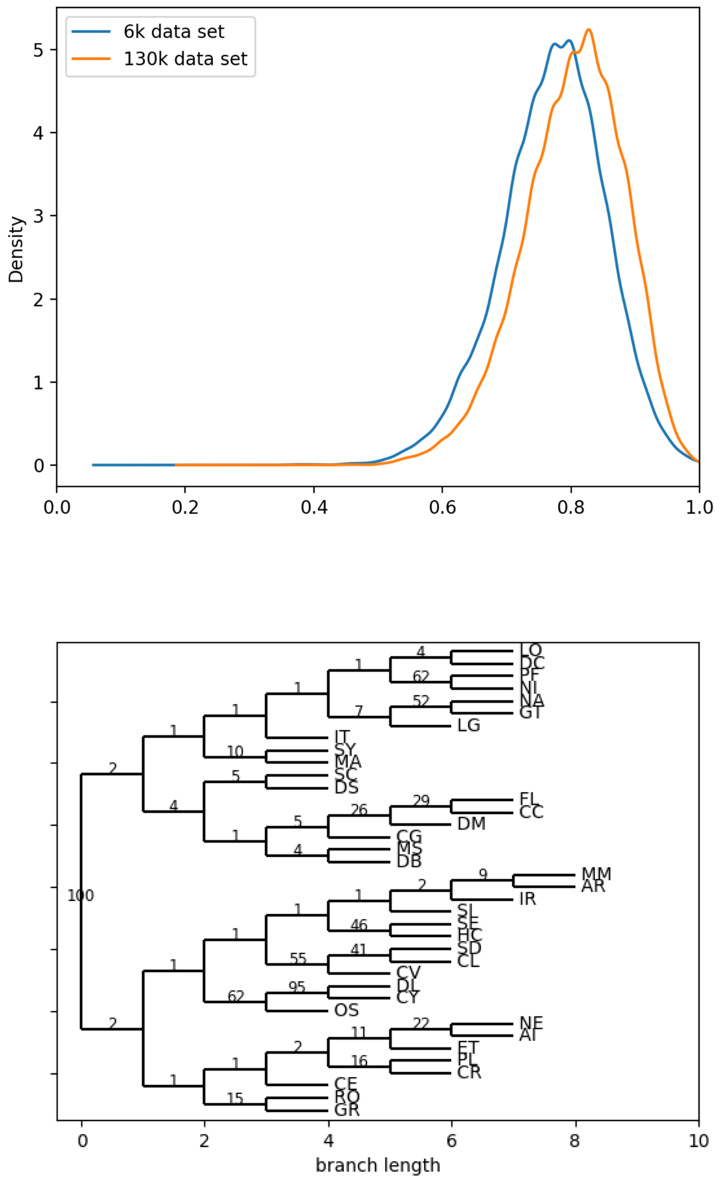


Figure 13: Top: kernel density estimates of all pairs from the 100 trees for the 60k data set and the 130k data set, respectively. Bottom: An example of the branch support result for a tree derived from the 130k data set. Acronyms correspond to categories in arXiv; see the corresponding full names in Table 3.

Acronym	Fullname	Acronym	Fullname
AI	Artificial Intelligence	IR	Information Retrieval
CC	Computational Complexity	IT	Information Theory
CG	Computational Geometry	LG	Machine Learning
CE	Computational Engineering, Finance, and Science	LO	Logic in Computer Science
CL	Computation and Language	MS	Mathematical Software
CV	Computer Vision and Pattern Recognition	MA	Multiagent Systems
CY	Computers and Society	MM	Multimedia
CR	Cryptography and Security	NI	Networking and Internet Architecture
DB	Databases	NE	Neural and Evolutionary Computation
DS	Data Structures and Algorithms	NA	Numerical Analysis
DL	Digital Libraries	OS	Operating Systems
DM	Discrete Mathematics	OH	Other
DC	Distributed, Parallel, and Cluster Computing	PF	Performance
ET	Emerging Technologies	PL	Programming Languages
FL	Formal Languages and Automata Theory	RO	Robotics
GT	Computer Science and Game Theory	SI	Social and Information Networks
GL	General Literature	SE	Software Engineering
GR	Graphics	SD	Sound
AR	Hardware Architecture	SC	Symbolic Computation
HC	Human-Computer Interaction	SY	Systems and Control

Table 3: Acronyms denoting the subject categories in arXiv.

support computation is shown here, for the 200 graphs are all similar and hence are omitted for brevity. As a result, the phylogram in Figure 13 is a tree for the 130k data set with branch support values obtained from the comparison to all the other trees for the same data set. The rest of the results in branch support also shows a similar pattern: most of the branches barely repeat, hence are again omitted.

The density estimates (Figure 13) can be interpreted as a smoothed version of a histogram. Each curve depicts the distribution of 4950 distances between 4950 pairs of trees, that is, all the possible combinations of the 100 trees for a data set. The curves almost overlap, both concentrating at a single point, around 0.8 of RF distance. The high distance values indicate that using the same sample is not helpful in obtaining stable results; topic models are too stochastic to produce a stable mapping.

On the other hand, the result from the branch support computation reveals some local structures that are successfully reproduced. In general, most bipartitions occur only once or twice in the other trees. Branch nodes that are direct ancestor of the leaves are more frequent: the branch support values range from 4 to 95, mostly greater than 10. However, the numbers decrease rapidly the closer they get to the root. It demonstrates the local structure is more conserved than the global structure. This pattern is shared among all the branch support results. It also explains why the RF distances are high but never reach 1.0. Several bipartitions are assured to be found in both of the trees, but those are too few compared to all the clusters in the whole tree.

The above demonstrates that the stochastic nature of the topic models is problematic despite using the optimal number of topics based on the coherence score.

6.3 Effect of Ensembling (RQ3)

The last experiment shows the effectiveness of ensembling. The effect is tested by adding the number of base models used in the ensemble to see how the differences between trees would change. The heatmap in Figure 14 shows the relationship between the Robinson-Foulds distance, where there are 1000 observations for each size. The blue line plots the average distances for each N.

Both the curve and the heatmap present a decreasing trend in RF distance. The trend almost asymptotes before it will intersect the horizontal axis, indicating ensembling more trees may not have much additional impact on consistency. The curve is monotonic, which means adding more trees within the range can always improve the performance. Overall, the dark coloured part of the heatmap is rather concentrated.

There is a clear "belt" of frequent distances. The performance of single models is presented at N=1. The distances are centered at 0.8, which indicates single models are mostly unstable. The belt loses its shape between 5 and 30 models, which shows the range when the ensemble performance varies greatly. Ensemble models whose size is greater than 30 are more stable, where more than 80% of the comparisons gives a distance around 0.1. Given that Robinson-Foulds distance is a strict metric, this is a major improvement for the mapping method.

6.4 Domain Structure in the Reconstructed Tree

Figure 15 shows the "best" tree this work has produced for Computer Science. Vectors used in the clustering are generated from the ensemble model that averages the prediction of the 100 topic models for the 130k data set. It is "best" because it utilises all the data and base models available in this work, which, according to the inferences I draw from the last two experiments, is the best approach to create a consistent model.

The topmost split in Figure 15 presents a similar pattern as in Huang et al. (2019), which distinguishes the applied and the theoretical disciplines. The

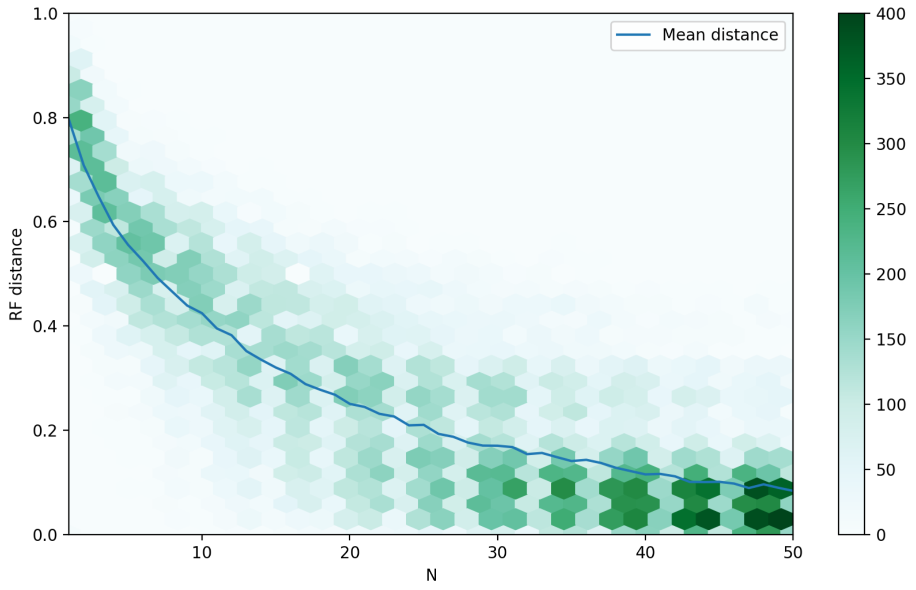


Figure 14: A heatmap for RF distance between ensembled topic models for 130k data set. N refers to the number of models fused in ensemble. The curve shows average distances for each N.

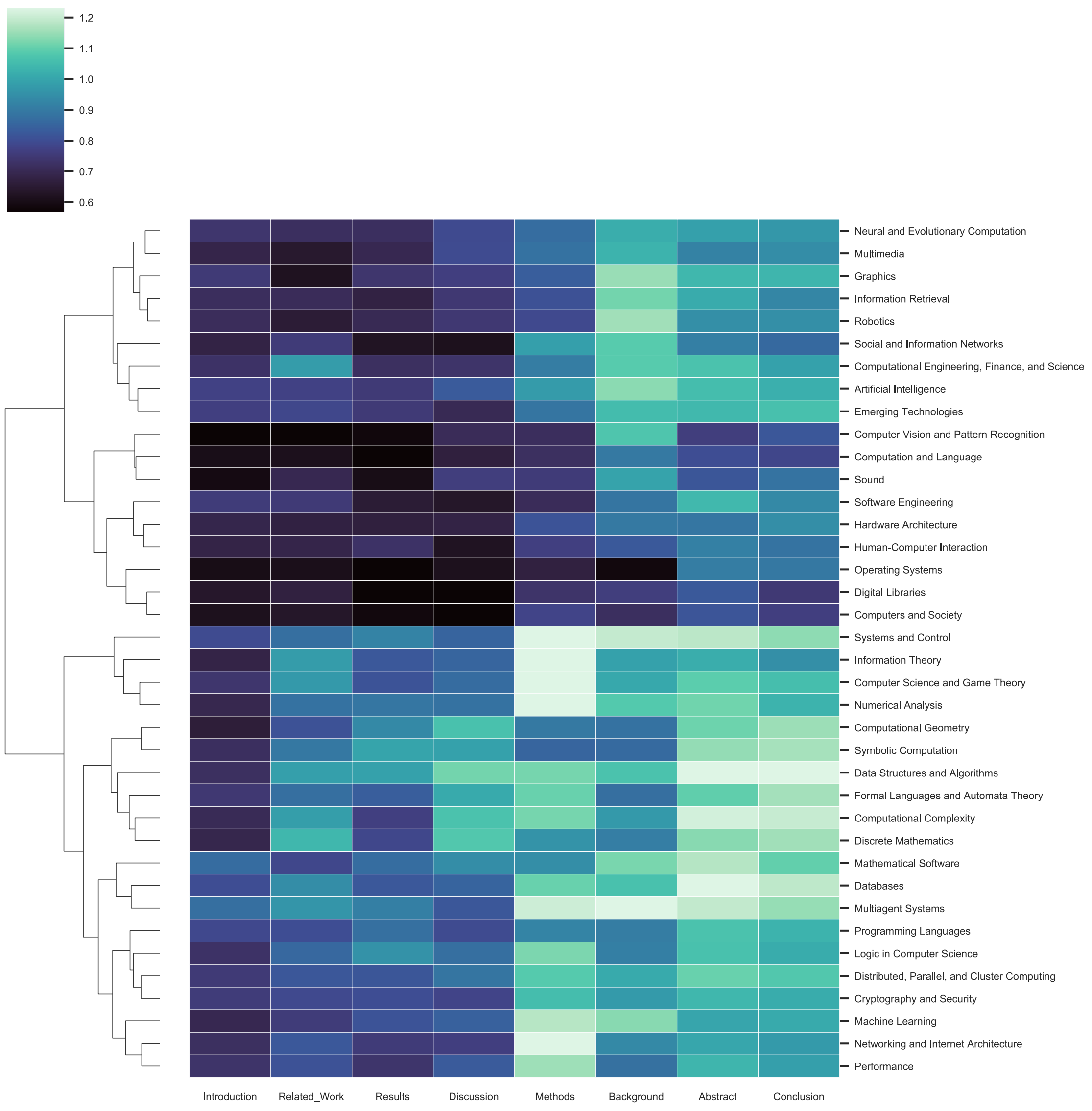


Figure 15: A heatmap and dendrogram showing the relatedness of subfields in Computer Science. The vectors are generated from the ensemble model that averages the prediction of the 100 topic models for the 130k data set. Numbers by the colour bar correspond to KL divergence in bits.

applied group (the upper sub-tree) generally have lower KL divergence in every section except the introduction, whereas the background, abstract and conclusion are less distinctive in the sense that the corresponding leaves are in slightly lighter colours and are closer to the sections in the theoretical disciplines. Overall, all the subjects write introductions that well summarise the full-text, which is indicated by the low KL divergences to the distribution of the full-text. The least representative section for the applied disciplines is background, and it is methods for the theoretical group.

Some groupings are surprising. For example, the direct node that contains Systems and Control connects it with seemingly unrelated fields, i.e. Information Theory, Game Theory and Numerical Analysis. The lighter colour for the methods section indicates that the methods deviates from the full-text to a greater extent than the other sections.

Some writing styles can be sensed from articles in Systems and Control. Some (n=30) articles in this area tend to require long proofs as the method chapter, which breaks the flow of the paper. The language in methods generally concerns mathematical derivations or algorithms to explain the controller of the system. Typical sections in this field include "problem formulation", "simulation results" and "proofs". Such structure is also found in papers from Information Theory, and less surprisingly, Game Theory.

Among the others, the nodes for Hardware Architecture and Operating Systems also seem unnatural, but this could be due to fewer articles. The observation aligns with wider confidence intervals, or equivalently, the higher variances in Figure 16, which shows mean KL divergence between topic distributions from abstract and full-text document. The tree is overall reasonable, while the unnatural groupings can be explained by the biases stated above.

7 Conclusion and Future Work

In this work, I explored the reproducibility of the science mapping method in Huang et al. (2019) from the perspective of sample randomness and algorithm randomness. Given that the exact data were not accessible, I did not repeat the scheme exactly as in the reference study, but instead used a much larger and more representative set of data to obtain new results. The new results were not concordant with the previous result. The set of maps (trees) produced from the newly obtained models showed inconsistent structures, despite some local branches within the trees were conserved. The original method in Huang et al. (2019) is found to be not robust and therefore needs

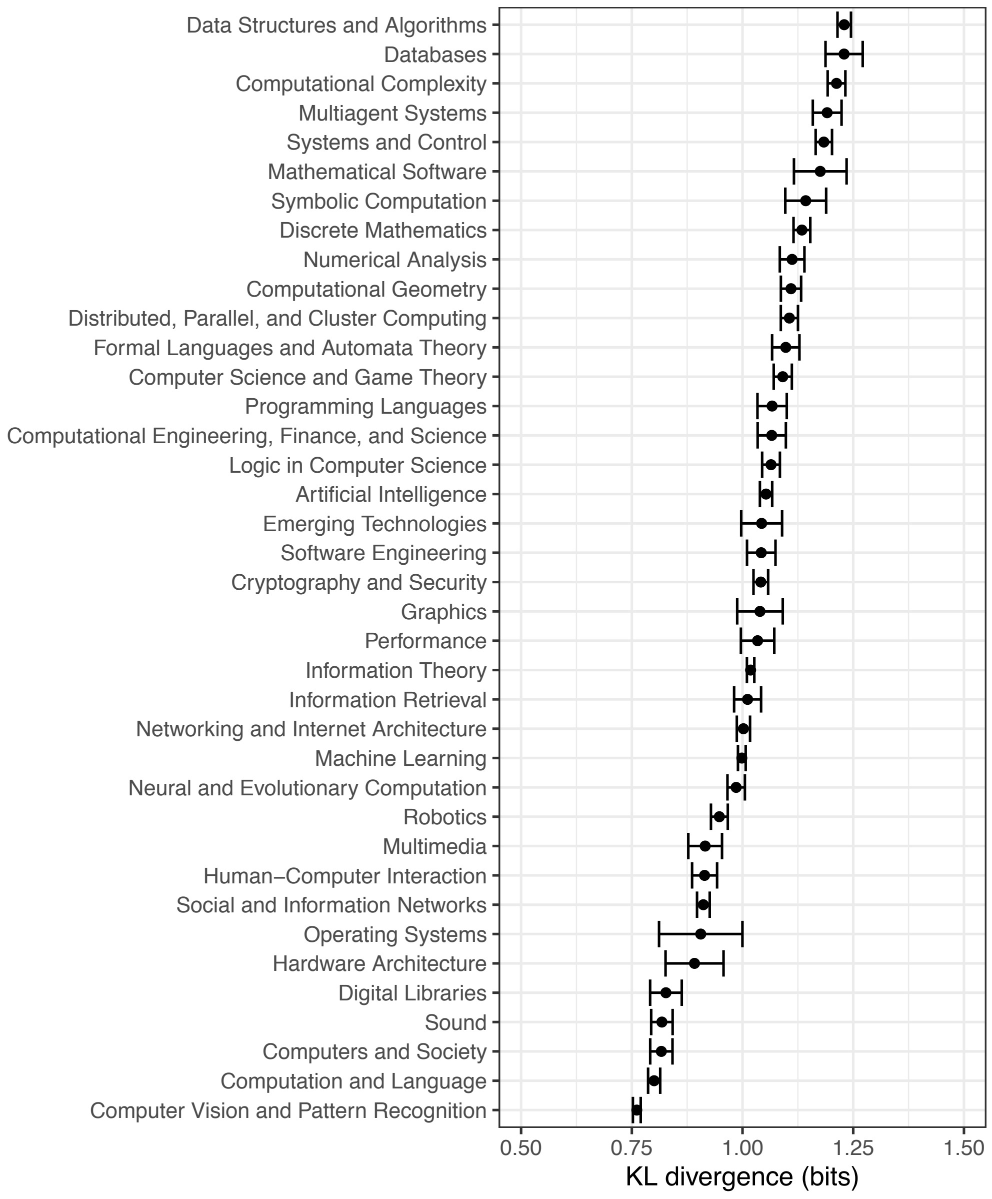


Figure 16: Mean KL divergence between topic distributions from abstract and full-text document. Error bars show 95% confidence interval. The names of the disciplines correspond to arXiv categories.

improving.

The major difficulty in reproducing the method in Huang et al. (2019) lies in the lack of details for the algorithmic settings. As the findings reveal, the variation in the result is mostly caused by the stochastic nature of LDA models. The settings used to train the models in this work are decided based on topic coherence scores. It is unclear whether Huang et al. (2019) used the same evaluation metric for tuning the number of topics, and the same is for the other settings such as the prior belief of the Dirichlet distributions. It seems the settings significantly impact the performance of topic models. As a consequence, the decisions made during tuning the settings affect how much variation is caused by randomness of the algorithm. In general, the mapping method is applicable. However, the reproducibility of the method can be affected by the specifics in the algorithm settings.

The research questions are answered as follows.

RQ1 Does the method from Huang et al. (2019) produce consistent results?

A1 Both consistency and inconsistency are found in the results. The lower-level structures are consistent, while the overall structures are not. Without ensembling, the method is not stable.

RQ2 What are the main sources of variation in the algorithm and the data set?

A2 In the algorithm: the stochastic nature of topic modeling.

In the data set: the imbalanced distribution of category papers, despite the impact it has on the overall consistency of models is slight.

RQ3 What improvements can be made to the method?

A3 Use the ensemble technique. It is effective to train multiple models and then average the resultant vectors to get a final result. It is recommended to train at least 40 models for the base, as in my case the effect of improvement reached the highest when that many models were used. However, the effect of more models can be limited after a critical value, since the effect of improvement dropped rapidly after adding enough models.

The method of interest can still be improved in many ways. A future work can improve the method by focusing on better use of topic models, for example if other techniques in machine learning would improve the consistency of the results. There are few studies on choosing the best presenting structures for the representation vectors; a investigation of visualising methods would be useful. As scientific maps are usually made for human readers to understand domains, it would be beneficial to have a systematic way to check the alignment with expert knowledge.

References

Srinivas Aluru. *Handbook of computational molecular biology*. Chapman and Hall/CRC, 2005.

Lars Arvestad. A simple utility for comparing phylogenetic trees, August 2010. URL `https://libraries.io/pypi/robinson-foulds?number=1.1`.

arXiv staff. About arxiv, 2004. URL `https://arxiv.org/about`. [Online; accessed 31-Dec-2019].

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55 (4):77–84, 2012.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Katy Börner. Analyzing and communicating the structure and evolution of science. *History and Philosophy of Science Talk, IUB*, 2004.

Katy Börner, Richard Klavans, Michael Patek, Angela M Zoss, Joseph R Biberstine, Robert P Light, Vincent Larivière, and Kevin W Boyack. Design and update of a classification system: The ucsd map of science. *PloS one*, 7(7), 2012.

Kevin W Boyack. Using detailed maps of science to identify potential collaborations. *Scientometrics*, 79(1):27, 2009.

Kevin W Boyack and Richard Klavans. Creation of a highly detailed, dynamic, global model and map of science. *Journal of the Association for Information Science and Technology*, 65(4):670–685, 2014.

Kare Bremer. Branch support and tree stability. *Cladistics*, 10(3):295–304, 1994.

Michel Callon, Jean Pierre Courtial, and Francoise Laville. Co-word analysis as a tool for describing the network of interactions between basic and technological research: The case of polymer chemsitry. *Scientometrics*, 22 (1):155–205, 1991.

Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon.

Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 03 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp163. URL `https://doi.org/10.1093/bioinformatics/btp163`.

Rucha Hareshkumar Dalwadi. Analyzing session laws of the state of north carolina: An automated approach using machine learning and natural language processing. 2020.

Sara Dolnicar and Friedrich Leisch. Evaluation of structure and reproducibility of cluster solutions using the bootstrap. *Marketing Letters*, 21(1):83–101, 2010.

J Fan and K Stewart. Detecting spatial patterns of natural hazards from the wikipedia knowledge base. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(4):87, 2015.

Belver C. Griffith, Henry G. Small, Judith A. Stonehill, and Sandra Dey. The structure of scientific literatures ii: Toward a macro- and microstructure for science. *Science Studies*, 4 (4):339–365, 1974. doi: 10.1177/030631277400400402. URL `https://doi.org/10.1177/030631277400400402`.

Gregor Heinrich. Parameter estimation for text analysis. Technical report, Technical report, 2005.

Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

Gerald James Holton. *Einstein, history, and other passions: The rebellion against science at the end of the twentieth century.* Harvard University Press, 2000.

Chien-yu Huang, Arlene Casey, Dorota Głowacka, and Alan Medlar. Holes in the outline: Subject-dependent abstract quality and its implications for scientific literature search. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, pages 289–293. ACM, 2019.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

Bruce R. Miller. Latexml: A latex to xml/html/mathml converter. https://dlmf.nist.gov/LaTeXML/, 2019.

Frank Nielsen. *Hierarchical Clustering*, pages 195–211. 02 2016. ISBN 978-3-319-21902-8. doi: 10.1007/978-3-319-21903-5_8.

Linda B Nilson and Ludwika A Goodson. *Online teaching at its best: Merging instructional design with teaching and learning research*. John Wiley & Sons, 2017.

Michael Paul and Roxana Girju. Topic modeling of research fields: An interdisciplinary perspective. In *Proceedings of the International Conference RANLP-2009*, pages 337–342, 2009.

HPF Peters and Anthony FJ Van Raan. Structuring scientific activities by co-author analysis. *Scientometrics*, 20(1):235–255, 1991.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

D.F. Robinson and L.R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131 – 147, 1981. ISSN 0025-5564. doi: https://doi.org/10.1016/0025-5564(81)90043-2. URL http://www.sciencedirect.com/science/article/pii/0025556481900432.

Henry Small. Visualizing science by citation mapping. *Journal of the American society for Information Science*, 50(9):799–813, 1999.

Munirathnam Srikanth and Rohini Srihari. Biterm language models for document retrieval. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 425–426, 2002.

Arho Suominen and Hannes Toivanen. Map of science with topic modeling: Comparison of unsupervised learning and human-assigned subject classification. *Journal of the Association for Information Science and Technology*, 67(10):2464–2476, 2016.

Charles Sutton and Linan Gong. Popularity of arxiv. org within computer science. *arXiv preprint arXiv:1710.05225*, 2017.

Jian Tang, Zhaoshi Meng, Xuanlong Nguyen, Qiaozhu Mei, and Ming Zhang. Understanding the limiting factors of topic modeling via posterior contraction analysis. In *International Conference on Machine Learning*, pages 190–198, 2014.

Hanna M Wallach, David M Mimno, and Andrew McCallum. Rethinking lda: Why priors matter. In *Advances in neural information processing systems*, pages 1973–1981, 2009.

Michael Waskom, Olga Botvinnik, Paul Hobson, John B. Cole, Yaroslav Halchenko, Stephan Hoyer, Alistair Miles, Tom Augspurger, Tal Yarkoni, Tobias Megies, Luis Pedro Coelho, Daniel Wehner, cynddl, Erik Ziegler, diego0020, Yury V. Zaytsev, Travis Hoppe, Skipper Seabold, Phillip Cloud, Miikka Koskinen, Kyle Meyer, Adel Qalieh, and Dan Allan. seaborn: v0.5.0 (november 2014), November 2014. URL `https://doi.org/10.5281/zenodo.12710`.

A Source code

A python script to extract Newick format trees from .csv files.

```
"""Get Newick trees from vector csv files

Usage:
python3 getnewick.py [path_to_csv] [dst_path]"""

import sys
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster import hierarchy
from os.path import join

def getDf(fpath):
"""Read csv files for vectors into pd.dataframe

Format for csv:

name,Abstract,Introduction,Background,Related_Work,
    Methods,Results,Discussion,Conclusion
Computational Complexity
    ,1.50,0.93,1.24,1.39,1.30,0.83,1.30,1.47
Logic in Computer Science
    ,1.45,1.00,1.26,1.25,1.41,1.26,1.21,1.39

...

"""
    df = pd.read_csv(fpath)
    df = df.set_index('name')
    df = df.sort_index()
    return df

def getNewick(node, newick, leaf_names):
    if node.is_leaf():
        return "%s%s" % (leaf_names[node.id], newick)
    else:
        if len(newick) > 0:
            newick = ")%s" % newick
        else:
            newick = ");"
        newick = getNewick(node.get_left(), newick,
           leaf_names)
        newick = getNewick(node.get_right(), ",␣%s" % (
           newick), leaf_names)
        newick = "(%s" % (newick)
        return newick

def getCluster(df):
    sns.set()
    sns.set(font_scale=0.6)
    g = sns.clustermap(df, metric='euclidean', method='
       complete', linewidths=.25, col_cluster=True,
       square=True, cmap="mako", robust=True)
    return g

def getTree(g):
    Z = g.dendrogram_row.linkage
    tree = hierarchy.to_tree(Z,False)
    leaveslist = hierarchy.leaves_list(Z)
    return tree, leaveslist

def getLeafnames(df, leaveslist):
    leaf_names = {leafid:df.index[leafid] for leafid in
        leaveslist}
    return leaf_names

def saveNewicktree(src, dst):
    df = getDf(src)
    g = getCluster(df)
    tree, ll = getTree(g)
    leaf_names = getLeafnames(df,ll)
    newick = getNewick(tree, "", leaf_names)
    with open(dst,'w') as f:
        f.write(newick)
    plt.close('all')
```

```
def computeNewick(df):
    g = getCluster(df)
    tree, ll = getTree(g)
    leaf_names = getLeafnames(df,ll)
    newick = getNewick(tree, "", leaf_names)
    return newick

if __name__ == "__main__":
    src = sys.argv[1]
    dst = sys.argv[2]
    saveNewicktree(src, dst)
```