Phrase indexing and the identification of related academic research content

Alexander Powell

27 April 2020 (revised 22 June 2020)

E-mail address for correspondence: ap.cybercraft@googlemail.com
ORCiD: https://orcid.org/0000-0001-6911-1890

Abstract

Work to automate the identification of related articles in corpora of academic research content is described. Pairs of related articles are recognised on the basis of the phrases they contain, using a similarity measure that emphasizes the importance of phrase overlap. Phrases are weighted according to their significance, evaluated in terms of statistical under- or over-representation relative to corpus-level frequency, and the significance scores of *n*-grams with higher *n* values are boosted. The measure proves broadly effective at identifying meaningfully related pairs of content items and may provide a useful basis for the development of 'see also'-type functionality.

Keywords:

phrase indexing, document similarity, document relatedness, Jaccard coefficient, TSW-OM similarity measure

1. Aims and context

The work reported here forms part of a project to create a simple PDF management tool, to meet the author's need for a mechanism to navigate a collection of several thousand locally stored PDF versions of academic research articles and assorted other content items, e.g. web pages and blog posts saved as PDF files. Building on previous unpublished work, the starting point is to index the 'bag of words' (actually a mixed bag of single words, bigrams and higher *n*-grams) extracted from each file in the PDF collection. A Windows GUI application has been developed to view and navigate around the resulting phrase indexes. It is possible to browse the indexed phrases, select a phrase, and then view a listing of the files in the collection that contain the selected phrase. (Phrase search functionality will be added at a later stage.) The user can then click on a filename and view the phrases extracted from it. Reciprocally, by clicking on a specific item in the list of phrases for the selected file it is possible to view the files that contain the phrase.

As an extension to the index navigation application, a further goal is to develop and implement 'see also' functionality, so that when a user selects a particular file, the application suggests and provides links to related files in the indexed corpus. This kind of functionality, now widespread and familiar, is one outcome of the half a century of fundamental research into information retrieval to which an extensive literature attests (e.g., in relation to the present investigations, refs 1-9). Increasingly, such functionality is delivered by AI-based approaches such as those developed by, for example, UNSILO (https://unsilo.ai) and Yewno (https://www.yewno.com). Typically the claim is made that these approaches go beyond the literal terms that occur in documents to identify the concepts they relate to. However, it is not always clear what their greater sophistication gives users, in terms of improved results or deeper semantic insight. Are the results obtained 15% better – assuming that having defined 'better' one can then quantify the degree of betterment – than those delivered by more rudimentary methods, or 150%? It would be helpful if we had more experience of working with, and greater practical understanding of the limitations of, the more rudimentary methods, so that the 'value add' of the AI-based approaches was more apparent. And in any case the potential of even quite simple approaches to text analysis and information retrieval remains for many publishers an under-explored area. The hope, therefore, is that this paper will assist others venturing into this terrain by describing in some detail one relatively straightforward approach to the determination of document relatedness.

2. Methods

Indexing

Phrase indexing software has been developed using Python, with guiding factors being speed of indexing, reasonable compactness of the resulting index structures, the possibility of indexing on an incremental (rather than all-in-one-go) basis, and the generation of index structures capable of supporting rapid navigation and retrieval of information relating phrases to files and vice versa. An additional aim was to have to undertake as little data preparation and cleansing as possible.

The indexer iterates recursively through a specified folder and its sub-folders, indexing any PDF files it encounters. The *pdftotext* executable from the *Xpdf* suite[1] is used to extract the text from each PDF file, with the 'nopgbreak' option specified. Phrases in the text are identified via the following steps:

(i) The lines of text are broken at punctuation and bracketing characters into lists of clauses (re.split('[.?!,;:/\(\)\[\]]', line).

(ii) Each clause is cleansed to ensure that retained clauses contain only alphanumeric characters, spaces and hyphens, and is then lowercased.

(iii) Each clause is split on spaces to yield a list of words (the_words). Hyphenated terms are treated as single words.

(iv) The word list is then processed to generate a list of phrases (a phrase here being a single word, bigram, trigram, or 4-gram):

```
for start in range(0, len(the_words)):
    if is_stop(the_words[start]) == False:
        for end in range(start, len(the_words)):
            phrase = the_words[start:end+1]
            # Block various stop phrases:
            if not (phrase in stop_phrases):
```

[1] https://www.xpdfreader.com

```
if len(phrase) == 1:
    W = phrase[0]
    # Don't allow stop words in single-word phrases:
    if ((bad_chars(W) == False) and (is_uni_stop(W) == False)
    and (is_stop(W) == False) and (len(W) > 1)
    and (not (w_digits_only(W)) or is_year(W) == True)  ):
        phrases.append(phrase)
elif len(phrase) < 5:   # 4-word limit
# Discard phrases having a terminal stop word:
    if is_stop(phrase[len(phrase)-1]) == False:
        stop_count = 0
        num_count = 0
        single_char_count = 0
        bad_word_count = 0
        for W in phrase:
            if bad_chars(W) == True:
                bad_word_count += 1
            if len(W) == 1:
                single_char_count += 1
            if is_stop(W) == True:
                stop_count += 1
            if w_digits_only(W) == True:
                num_count += 1
# Some more filtering: exclude phrases containing too many stop words,
# too many numbers, any strings that don't include at least one letter or
# number ('bad' words), or composed entirely of single-character words:
        if ( (bad_word_count == 0) and (stop_count < 2) and (num_count <
        len(phrase) and single_char_count < len(phrase) ):
            phrases.append(phrase)
```

There are separate lists of stop words for single words (unigrams) and words occurring in n-grams for which $n > 1$ (see **Supplementary Materials C, Appendix B**); these are utilized by the is_uni_stop and is_stop procedures respectively, referred to in the code above.

(v) The phrases are sorted, and duplicates counted.

(vi) If a phrase occurs in a file more than C times then it is added to the list of retained phrases. The value of C can be adjusted; a value of 1 was used in the work reported here, i.e. phrases had to occur twice or more. Using this value for C, the phrase list is reduced in size to around 10% of the C=0 size.

The list of retained phrases is fed directly into the indexing subroutines, no stemming or other language processing operations being performed. The indexing process generates and updates a number of inter-related resources:

files.dat: a text file containing the names of indexed files, their word counts (tokens), and the start offsets of the first and last records in file-phr.dat (see below) that relate to each file.

corpus.dat: a binary file storing information about the corpus.

phrases.idx: a binary file associating phrase ID values with their phrases.

global-phr.dat: a binary file containing fixed-length records of data about each phrase in the corpus, e.g. total no. of instances of a phrase, no. of files containing a phrase. Phrase records are written to the file in the order in which the phrases occur in the files. If a corpus file contains a phrase for which a record has already been written to *global-phr.dat*, a new record for the phrase is not created; instead the token count for the phrase is updated in the existing record.

file-phr.dat: a binary file containing fixed-length records of data about the instances of each phrase occurring in each file of the corpus, e.g. the number of phrase instances that occur in the file. Each file in the corpus is represented in file-phr.dat by a block of contiguous records. A record is created only the first time a phrase is encountered in a file; subsequent mentions of the phrase *in the same file* just update the existing record's token count.

While file-phr.dat grows approximately linearly in relation to the quantity of text indexed, global-phr.dat grows increasingly slowly, as the likelihood that a phrase has already been encountered rises as more content is indexed.

It is not an objective of the project to develop a full-text search engine capable of identifying the exact locations where particular terms occur, so the byte offsets of terms occurring in the corpus files are not recorded.

The corpus

The test corpus is a diverse set of 257 PDF files representative of the author's collection. It includes several clusters of articles relating to specific areas of academic research, including protein folding and dynamics, autism, collective intentionality, and consciousness; it contains several complete books; and it includes a variety of closely related content items, such as an article in two parts about Cambridge biochemist Frederick Gowland Hopkins, a number of obituaries, and a trio of reviews of the same book.

Topical relationships

Analysis of file relationships is performed by custom Python subroutines which take the phrase indexes as inputs. The similarity measure employed owes its form to two powerful intuitions long familiar to others working in the area: (1) that similar documents will tend to have more phrases in common than will dis-similar documents, and (2) that not all phrases should weigh equally in the assessment of similarity. If we consider a pair of document files A and B, then apropos (1) attention focuses on the ratio of the number of phrases file A shares with file B to the number of phrases in file A but not in file B, and similarly on the ratio of the number of phrases file B shares with file A to the number of phrases in file B but not in file A. The numerator in both these ratios is of course just the intersection of the set of file A phrases and the set of file B phrases, and intuitions about the importance of phrase overlap between similar files would be gratified (it was felt) if the ratios pertaining to the two files were multiplied together to obtain an overall measure of similarity. Hence if $OM_{A,B}$ (OM standing for Overlap Multiplication) is the similarity of a pair of files A and B then

$$OM_{A,B} = N_{AB}^2 / (N_A \times N_B) \qquad \textbf{(1)}$$

where N_{AB} is the number of phrases that occur in both files A and B, N_A is the number of phrases found in A only, and N_B is the number of phrases found in B only. It should be noted that 'number of phrases' here refers to distinct *types*, not *token* counts. The value of *OM* will be high in cases where the intersection size is large relative to the number of phrases unique to each of the files in the pair under consideration, and low when the intersection size is small relative to the numbers of unique phrases.

There is something of a resemblance, albeit only as regards broad form, between this measure and the set theoretic quantity known as the Jaccard index or similarity coefficient, which may be expressed thus:

$$J_{A,B} = |A \cap B| \;/\; |A \cup B| \qquad \textbf{(2a)}$$

or

$$J_{A,B} = |A \cap B| \;/\; (|A| + |B| - |A \cap B|) \qquad \textbf{(2b)}$$

Indeed the Jaccard coefficient is well-known in relation to the measurement of document similarity [1]. The difference between it and (**1**) is clear enough: (**1**) will tend to amplify the effects of intersection size relative to the Jaccard measure.

The second intuition about document similarity mentioned above was to do with the weighting of phrases on the basis of their topical importance or significance. In the present work a phrase is deemed to be significant in a file if it occurs more frequently in the file than one would expect given how frequently it occurs in the corpus overall, assuming naively that phrases are distributed uniformly throughout the corpus. Thus the significance of a phrase occurring in a file is given by the expression:

$$sig = N_{\text{actual}} \,/\, N_{\text{expected}} \qquad \textbf{(3a)}$$

Now if we suppose that

$$N_{\text{expected}} = N_{\text{corpus}} \times C_{\text{file}} \,/\, C_{\text{corpus}} \qquad \textbf{(3b)}$$

where C_{file} is the word count of the file in question, C_{corpus} is the total word count of the entire corpus, and N_{corpus} is the number of times the phrase occurs in the corpus, then

$$sig = (N_{\text{actual}} \times C_{\text{corpus}}) \,/\, (C_{\text{file}} \times N_{\text{corpus}}) \qquad \textbf{(3c)}$$

The weighting scheme also incorporates a crude attempt to represent a third intuition: that higher *n*-grams, i.e. multi-word and hyphenated terms as opposed to unigrams, should be scored more highly, and increasingly so as *n* increases. The basis for this intuition is probabilistic: if phrases were formed by the random combination of words selected from a set, the chance of forming any specific chain of words would decrease with increasing chain length. Ceteris paribus a particular number of occurrences of a long word chain is less probable, and therefore more significant, than the same number of occurrences of a shorter word chain.[2] A trigram should therefore score more highly than a bigram, and a bigram should score more highly than a unigram. To implement this idea, the score for each phrase is multiplied by the total number of spaces or hyphens in the phrase string plus one.

[2] This way of thinking calls to mind Zellig Harris's theory of language, in which meaning inheres in departures from equiprobability in the combination of linguistic elements [10].

The process of discerning relationships between the files in the corpus involves the pairwise comparison of all the files. For a corpus of N files, there are ½($N^2 - N$) file pairs to consider (if we avoid comparing each file with itself). For the test set of 257 files, therefore, there are 32,896 file pairs to be analysed. The method used to compute the similarity score for a pair of files A and B can be summarised thus:

(i) Interrogate the phrase indexes to generate the list of phrases for each file in the corpus. The file corpus.dat (produced during indexing) stores the offsets in file-phr.dat of the first and last records for each file, so it is possible to access the relevant block of phrase records directly. Recall that file-phr.dat contains only *one* record for each phrase that occurs in a file.

(ii) A somewhat permissive filter is applied to prevent phrases that are excessively frequent at the corpus level from being added to a file's phrase list. Specifically, phrases are considered only if they occur in less than two-thirds of the corpus files, and if the total phrase count across the corpus is less than the corpus word count divided by pi times the number of files. (These filter parameter values were established empirically; pi has no special theoretically grounded significance but rather represents a light-hearted approximation to three, which was found to work well as a multiplier.)

(iii) The phrase list obtained at this stage for each file (as opposed to at the prior indexing stage) actually consists of a dictionary of items in which the keys are phrase IDs and the values are the computed phrase significance scores (which are specific to the file in question). The significance scores are derived in accordance with Eqns. **3a**, **3b** and **3c**. Each file's phrase list is added as an item to a single overarching list (f_phr_list). The list item relating to a specific file can be accessed directly since its index in f_phr_list is just the file's ID value.

(iv) The pairwise file comparisons are carried out by a nested pair of for loops:

```
for A in range(0, max_file_id+1):
    # Get A's phrases...
    A_dict = f_phr_list[A]
    for B in range(A+1, max_file_id+1):
        # Get B's phrases...
        B_dict = f_phr_list[B]
        # And then compute the A-B pair strengths...
        (intersect, tot_sig) = find_intersection(A_dict, B_dict)
        A_only = (len(A_dict)-intersect)
        B_only = (len(B_dict)-intersect)
AB_score = (math.sqrt(tot_sig) * 100) * ( intersect**2 / ((1+A_only) * (1+B_only)) )
```

The call to the find_intersection subroutine returns the intersection of the two files' phrase lists (i.e. the number of phrases contained by both file A and file B)(intersect) and also the 'total significance' (tot_sig), this being defined as the product of (a) the average significance with respect to file A of the phrases in the intersection and (b) the average significance with respect to file B of those same intersectional phrases (see below).

(v) The use of a dictionary rather than a plain list to store the phrase ID / significance pairs makes the derivation of the phrase intersection between two files rather rapid, since only the list of file A phrases needs to be iterated over: the list of file B phrases can be queried directly using the file A phrase IDs as keys[3]:

```
def find_intersection(dictA, dictB):
# Return the elements that are common to dictA and dictB
#  Inputs are now dictionaries of the form {phr_id: signif, ... }
  intersect = []
  sum_A_sigs = 0.0
  sum_B_sigs = 0.0
  for key in dictA:    # key is a phrase ID
    sig1 = dictA[key]   # sig1: significance in file A of phrase with ID value = key
    if key in dictB:
      sig2 = dictB[key]   # sig2: significance in file B of phrase with ID value = key
      intersect.append( [key] )  # add to intersection if phrase is common to A and B
      sum_A_sigs = sum_A_sigs + sig1
      sum_B_sigs = sum_B_sigs + sig2
  # Compute the significance-based weighting by which to multiply the A-B link strength
  #    - the product of the average significance values for A and B:
  intersect_size = len(intersect)
  av_A_sig = sum_A_sigs / (1+intersect_size)
  av_B_sig = sum_B_sigs / (1+intersect_size)
  tot_sig = av_A_sig * av_B_sig
  # Return the intersection size and combined significance:
  return (intersect_size, tot_sig)
```

(vi) The overall similarity between files A and B is calculated as the product of an adjusted version of the total significance, taking into account phrase frequency statistics and the number of phrase components as outlined above, and the Overlap Multiplication (OM) term already discussed:[4]

```
AB_score = (math.sqrt(tot_sig) * 100) * ( intersect**2 / ((1+A_only) * (1+B_only)) )
```

The overall similarity measure computed in step (vi) is henceforth referred to, for reasons which are obvious enough, as the Tempered-Significance-Weighted Overlap Multiplication (or TSW-OM) method. Note that unity is added to the denominators in the OM term to avoid the possibility of division by zero (as would otherwise arise if files A and B were identical and hence there were no phrases lying outside the intersection). File relationships were also computed for the files in the test corpus using just the Jaccard coefficient pure and simple.

[3] Initially standard Python lists were used for storing each file's phrase ID/phrase significance pairs, and to find the intersection involved nested for loops to iterate over the files A (outer loop) and then, for each file A, over the files B (inner loop). Switching to the dictionary-based approach outlined above reduced the time taken to compute the similarities for the test corpus by very roughly an order of magnitude.

[4] The adjustment to the significance (involving square root and multiplication operations) was introduced pragmatically in order to temper the impact of the significance weighting on the overall measure, when initial experiments raised the possibility that its effect in combination with the OM term was perhaps excessive.

3. Results and discussion

Phrase indexing

Different combinations of maximum permitted phrase length and minimum permitted number of phrase occurrences were tried, using the TSW-OM similarity measure. As expected, admitting longer phrases increases the number of phrases indexed, as does reducing the number of instances required of a phrase in a file. Smaller numbers of indexed phrases are associated with smaller intersection sizes, and smaller intersection sizes lead to lower similarity scores.

Prima facie it would seem desirable to exclude as few phrases as possible from indexing, but requiring just a single instance of a phrase to occur in a file was found to result in excessively large index structures and sluggish index-processing operations. An acceptable trade-off, in terms of indexing and index-processing times, size of index structures, and number of phrases indexed, was found by experiment to be achieved by specifying a maximum phrase length of four and a minimum phrase token count per file of two. It was this '4-2' indexing scheme that was employed to generate the index structures underlying the file similarity computations.

Indexing in this way the 257-file test corpus, which amounts to nearly 2.9m words, takes several minutes on a fairly ordinary i7-powered laptop running Windows 10, and yields a global-phr.dat file of 2.93MB and a file-phr.dat file of 6.2MB. In total 149,803 distinct phrases are indexed. (See **Supplementary Materials B**.)

TSW-OM

Casual inspection of the computed file similarities derived from indexes generated using the 4-2 scheme and scored using the TSW-OM similarity measure suggests that the approach generally identifies rather well the kinds of file relationship that would be discerned by a knowledgeable human investigator. (See **Supplementary Materials A**). The top 20 corpus file pairs, when entries are presented in descending order of score, are illustrative. (See **Table 1**.)

In top place is the (74,167) pair, which corresponds to two slightly different PDF versions of the same article, saved from the Web, about protein folding. Also prominent are the clusters of content items to do with the work of Gallotti and Frith on collective intentionality (82,83,84), the passing of protein chemist Chris Dobson (166,232), and reviews of Powell's *Economic History of the British Building Industry* (45,86,248). It is gratifying to see Milne's obituary of physicist James Jeans in *Biographical Memoirs of Fellows of the Royal Society* paired strongly with the entry by Meadows for James Jeans in the *Oxford Dictionary of National Biography* (113,154); likewise the pairing of two *Scientific American* articles by cell biologist Mark Bretscher (31,32); and also a *Physics World* story about a development in protein folding research paired with the research article to which the story relates (172,173). The intersection phrases of selected document pairs are shown in **Appendix A** (see **Supplementary Materials C**).

The TSW-OM measure differentiates strongly between different degrees of document similarity. The 5th-ranked pair scores 3621.3, the 10th-ranked pair scores 1795.6, the 20th-ranked pair scores 967.04, the 40th-ranked pair scores 486.31, the 100th-ranked pair scores 183.36, the 200th-ranked item scores 115.18, and the 400th scores 71.83. Empirically and subjectively, it appears that a score of around 100 or so is required for the inference of a meaningful relationship between two documents to be reasonable. Such a threshold score is attained or exceeded by roughly the top 240 (0.73%) document pairs in the test corpus.

Table 1: Top 20 file pair matches resulting from application of TSW-OM measure against 257-file test corpus (4-2 indexing scheme).

Rank	File A	Filename A	File B	Filename B	Score
1	74	Everts_2017_Protein_folding.pdf	167	Protein folding_ Much more intricate than we thought _ July 31 2017 Issue - Vol. 95 Issue 31 _ Chemical & Engineering News.pdf	682613
2	82	gallotti_frith_di_paolo_et_al_we-mode-intentionality.pdf	83	gallotti_frith_REPLY_to_DiPaolo_et_al.pdf	63775.93
3	166	Professor Sir Christopher Dobson chemist whose work on proteins advanced research into Alzheimer's and Parkinson's disease – obituary.pdf	232	Tributes paid to Master of St John's College who has died age 69 _ StJohns.pdf	7300.668
4	172	Quantum approach reveals faster protein folding – Physics World.pdf	173	Quantum Approach to Fast Protein-Folding Time.pdf	3686.773
5	45	Cooney_1981_review_Powell_Econ_hist.pdf	86	Gardner_review_Powell_an_economic_history.pdf	3621.298
6	113	JamesJeans_obit_notice_RS.pdf	154	Oxford DNB article_ Jeans.pdf	2620.262
7	83	gallotti_frith_REPLY_to_DiPaolo_et_al.pdf	84	Gallotti_Frith_Social_cognition_in_we-mode.pdf	1964.417
8	17	Barbieri_M_2011_A_Mechanistic_Model_of_M.pdf	18	Barbieri_M_2013_The_Paradigms_of_Biology.pdf	1920.153
9	31	Bretscher_1985_Molecules_of_the_cell_membrane.pdf	32	Bretscher_1987_How_animal_cells_move.pdf	1823.012
10	246	Will the world embrace Plan S_Science_2019.pdf	253	Yes it is getting harder to publish in prestigious journals if you haven't already _ Science _ AAAS.pdf	1795.571
11	82	gallotti_frith_di_paolo_et_al_we-mode-intentionality.pdf	84	Gallotti_Frith_Social_cognition_in_we-mode.pdf	1653.886
12	238	Verd_et_al_2018_Drosophila_damped_oscillator.pdf	239	Verd_Monk_Jaeger_2019_Modularity_criticality_evolvability_dev_reg_network.pdf	1562.003
13	45	Cooney_1981_review_Powell_Econ_hist.pdf	248	Witham_Review_Powell_Economic_hist_book.pdf	1340.287
14	62	Dobson_etc_1999_Heterogeneous_folding_of_equine_lysozyme.pdf	63	Dobson_Evans_Radford_1994_Lysozyme_story.pdf	1281.507
15	86	Gardner_review_Powell_an_economic_history.pdf	248	Witham_Review_Powell_Economic_hist_book.pdf	1250.332
16	224	The_making_of_a_biochemist_II_The_construction_of_.pdf	225	The_Making_of_a_Biochemist_I_Frederick_Gowland_Hop.pdf	1177.229
17	110	IIT is scientifically falsifiable – Erik Hoel – Medium.pdf	233	TsuchiyaAndrillonHaun2019.pdf	1164.723
18	212	Tainer_Getzoff_et_al_1985_Atomic_mobility_antigenicity.pdf	243	Westhof_et_al_1984_Segmental_mobility_antigenicity.pdf	1009.134
19	72	empathizing_vs_systematizing.pdf	256	Zheng_Zheng_2017_empathizing-systemizing_mental_rotation.pdf	970.3263
20	24	Benkovic_et_al_2008_Free-energy_landscape_of_enzyme_catalysis.pdf	91	Goodey_Benkovic_2008_Allosteric_regn_and_catalysis_emerge_via_common_route.pdf	967.0399

Of particular interest are documents that fail to connect strongly with any other documents in the corpus, for example documents 99 and 205. The former is a playful piece from the *Veterinary Record* about the breeding of haggis (!)[11], while the latter is a curious article about cardiovascular disease that exhibits a remarkably high level of self-citation [12]. That these documents fail to pair strongly with others seems appropriate and reassuring; one wonders whether in a much larger and more diverse corpus they would still fail to find partners. If an aspect of their idiosyncrasy is an unusual *combination* of significant phrases then perhaps they would not. Whilst one phrase might associate such a document with one particular set of documents, another phrase might associate it with a different set, and another phrase might associate it with yet another. But to be associated strongly with another document typically requires the sharing of a number of significant phrases, and significant phrases tend to be limited in number. When a document's significant phrase relationships are divided between many distinct document clusters, isolation (one conjectures) will often be the result. That possibility suggests some interesting potential use cases for a methodology like that described, in the detection of articles that, for whatever reason, are epistemic outliers of some kind.

Comparison of TSW-OM with pure Jaccard measure

It is instructive to compare the above TSW-OM scores of selected ranked file pairs with those of their Jaccard equivalents. The Jaccard measure by definition yields scores of 1.0 or less. We find, in relation to the test corpus, that the 5th-ranked file pair scores 0.1997, the 10th-ranked pair scores 0.1729, the 20th-ranked pair scores 0.1546, the 40th-ranked pair scores 0.1426, the 100th ranked pair scores 0.1284, the 200th ranked item scores 0.1183, and the 400th scores 0.1097. The 400th ranked file pair therefore has a Jaccard score of 85% of that of the 100th-ranked pair, and 63% of that of the 10th-ranked pair, whereas its TSW-OM score is only 39% of that of the 100th ranked pair and just 4% of that of the 10th-ranked pair.

This difference between the TSW-OM and Jaccard scores becomes very apparent when document relationships within a corpus are visualized as a graph.[5] In **Figures 1 and 2** the weights of the lines connecting the document nodes reflect the pair similarity scores. **Figure 1** visualizes the relationships as evaluated using the pure Jaccard measure while **Figure 2** visualizes the relationships determined using the TSW-OM measure. The pure Jaccard measure fails to distinguish emphatically between strong and weak degrees of similarity, with scores being distributed fairly evenly over a large fraction of the theoretically possible interval. As a result, each node of the graph appears to be connected non-negligibly to a host of other nodes and it is hard to differentiate between strong and weak connections. The TSW-OM measure, in contrast, clearly distinguishes between different degrees of similarity. When one document node is selected and its relationships are highlighted, the strongest connections are readily apparent.

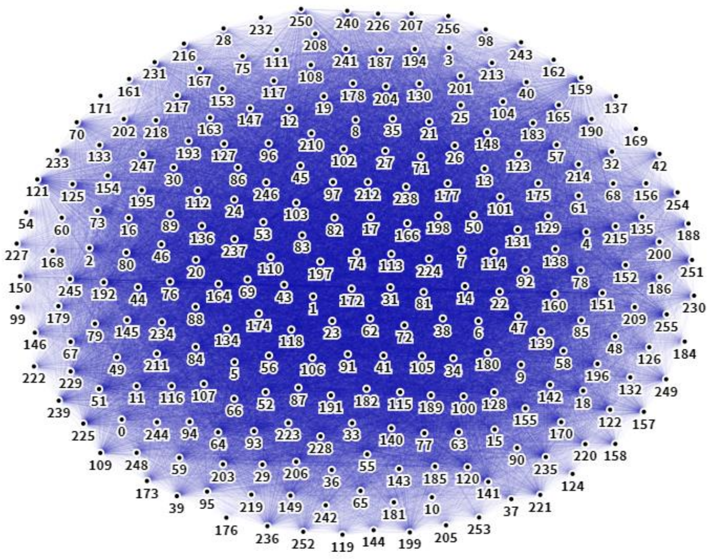
In addition to the issue of the degree of differentiation between scores, and the way the TSW-OM measure lifts the scores of highly related document pairs up above the welter of weaker connections, there is the matter of the rank ordering of similarity scores. How do the score orderings yielded by the two measures differ? The results spreadsheet provides the answers. (See **Supplementary Materials A**.) If variations of colour, text weight and highlighting are used to distinguish the top 200 (say) document pairs under each similarity scoring measure, it is possible to see (by sorting according to a particular measure's scores) how rank order varies under the different measures. It is clear that the ordering yielded by the Jaccard measure differs greatly from

[5] The Flourish online visualization tool (https://app.flourish.studio/) was used.

that given by the TSW-OM measure. For example, the 4th-ranked document pair under TSW-OM has a Jaccard score of just 0.0724, placing it in 3972nd place in the Jaccard rankings. This is a clear win for TSW-OM, since this pair consists of an article outlining a quantum approach to protein folding and a *Physics World* item summarizing that article. Similarly, documents 43 and 83 form a related pair – part of the collective intentionality cluster – but their Jaccard score is just 0.0578, which corresponds to a Jaccard ranking of 7922. Under TSW-OM their score of approximately 221 equates to a ranking of 79.

Figure 1: Visualization of document similarity using simple Jaccard approach. (**A**) Complete network; (**B**) document 81 selected and relationships highlighted. (Interactive figure at https://public.flourish.studio/visualisation/2074285/)

(A)



(B)

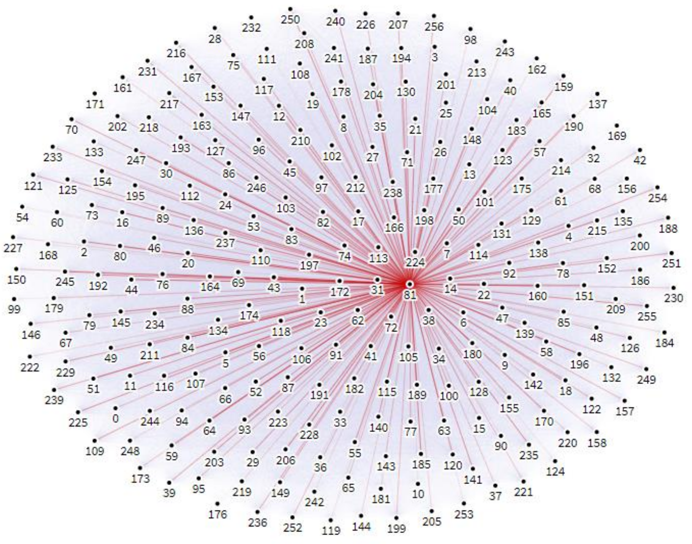
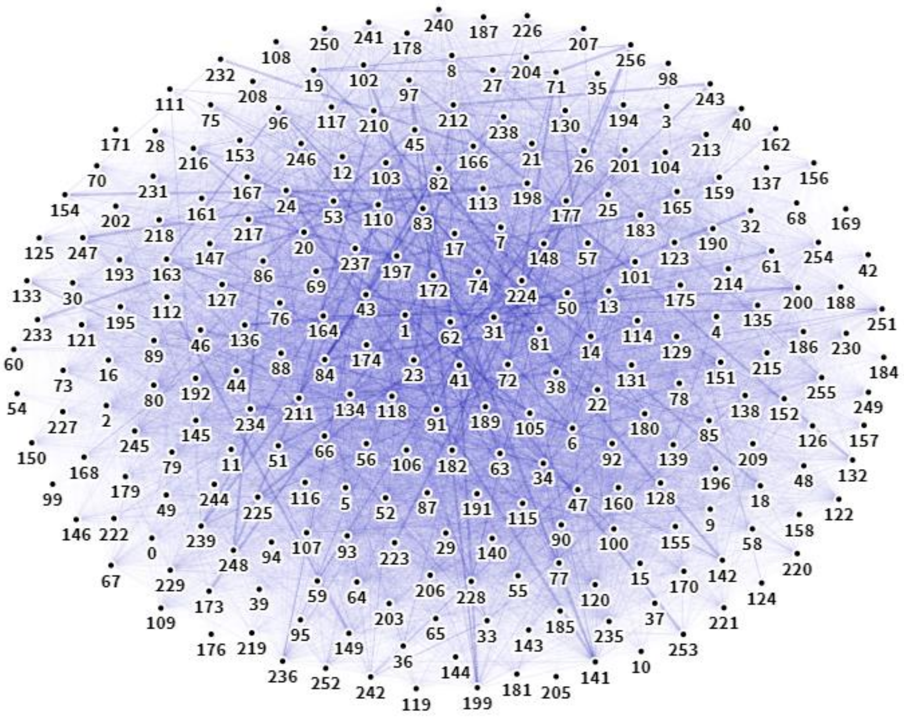
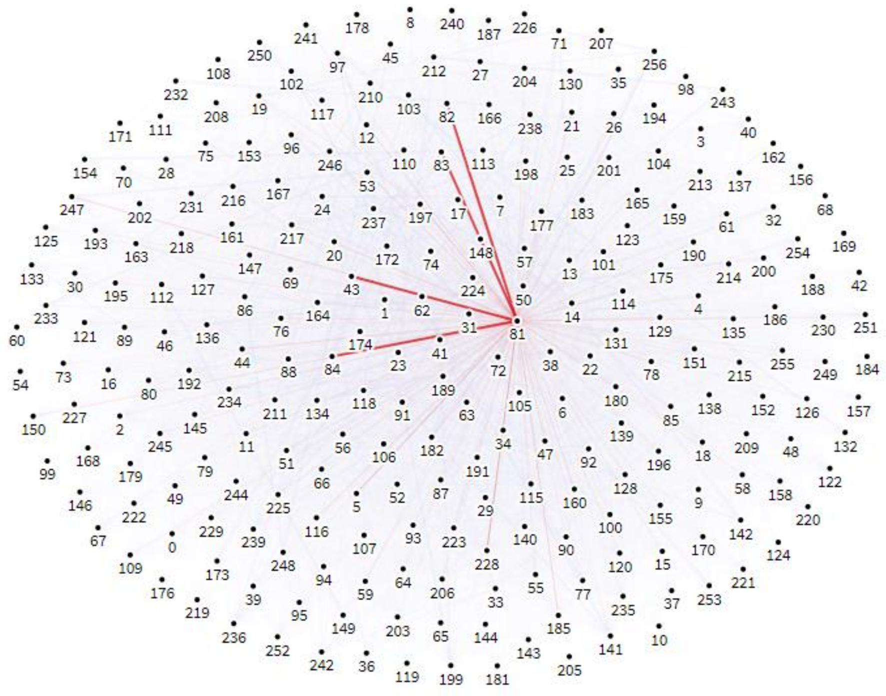


Figure 2. Visualization of document similarity using TSW-OM measure. (**A**) Complete network; (**B**) document 81 selected and relationships highlighted. (Interactive figure at https://public.flourish.studio/visualisation/2073932/)

(A)



(B)

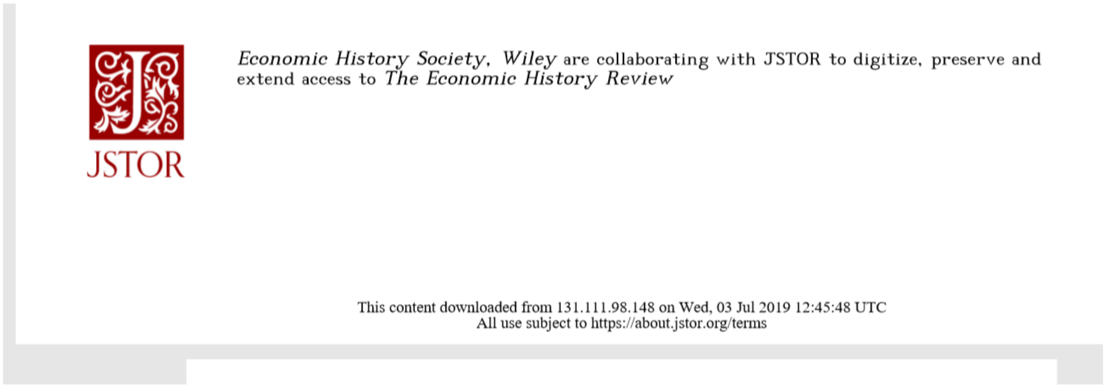


Artefacts and anomalies

Notwithstanding the numerous successes of the TSW-OM measure when compared with the Jaccard measure, oddities are occasionally seen. Typically these result from artefacts in the text extracted for indexing from the original PDF file. Some artefacts are removed by the filters applied during indexing. For example, prior to the addition of a filter to ignore phrases consisting entirely of single-letter words the (107,143) document pair gained 18th place ranking under the TSW-OM measure, with a score of over 1000. This was surprising, since document 107 is a *Nature* paper from the Baker lab about de novo protein design, while document 143 is a *Nature* news feature about the authorial practice of self-citation. The cause of the anomalously high similarity score was the presence in both documents of unusual trigrams consisting solely of single-letter words, seemingly derived from strings in the PDF file formed by the insertion of spaces between the characters of particular words. The addition of the single-character filter reduces the TSW-OM score for the (107,143) pair to a paltry 1.336, thereby lowering the ranking by some 27825 places.

PDFs derived from online magazines and blogs are frequently found to contain phrases relating to the identity of the publication, and saving documents by using the browser's 'print to PDF' option is liable to create headers and footers that may also contaminate the text fed to the indexer. Without effective counter-measures these factors typically lead to the attribution of excessively high similarity ratings to pairs of documents from the same publication, irrespective of their actual subjects. Some of the pairs of items from *Quanta Magazine* (https://www.quantamagazine.org/) were found to exemplify this problem, a fact which led to the introduction of stop phrases into the methodology. PDF artefact issues remain, however, as the cases presented in section III of **Appendix A** show. Of those, document pair (246,253), a pair of *Science* magazine stories (rather than research articles) that were saved by printing to PDF, is surely the most egregious. Their TSW-OM score of 1795.57 results from the shared incidence of strings 'aaas', 'careers', 'contact us', 'email address', 'emails', 'jobs', 'newsletters', 'privacy policy', 'receive emails', 'science aaas', 'sciencemag', 'sign up', 'subscribe', 'subscribe today' and 'table of contents'. The high TSW-OM score of document pair (45,139), meanwhile, is attributable almost entirely to strings occurring on the JSTOR cover page (**Figure 3**). Such cases suggest that further refinement of the list of stop phrases would pay dividends. Alternatively, or additionally, it may be necessary to validate terms against a dictionary and/or ruleset before committing them to the phrase indexes.

Figure 3. Portion of JSTOR cover page.


JSTOR

Economic History Society, Wiley are collaborating with JSTOR to digitize, preserve and extend access to *The Economic History Review*

This content downloaded from 131.111.98.148 on Wed, 03 Jul 2019 12:45:48 UTC
All use subject to https://about.jstor.org/terms

4. Concluding remarks

The main aim of the work reported here was to develop a simple, automated method for identifying related files in moderately sized PDF corpora. Initial analysis suggests that in realizing that aim it has been successful, inasmuch as the file relationships identified within the test corpus are, generally speaking, significant and meaningful when the similarity score exceeds a particular threshold value. However, the use of PDF files has implications for the form and quality of the text indexed, and this has the potential to distort results and throw up occasional, and sometimes serious, anomalies. If the methodology described were applied to HTML or XML versions of articles it seems likely that most of the anomalies encountered would be eradicated.

Ample scope exists for further analysis of the results reported here and of results obtained when the method is applied to other corpora of different sizes and compositions, and for comparison with the results generated by other similarity measures. In particular, it would be interesting to compare the weighting scheme employed in these investigations with the classic TF-IDF (term frequency—inverse document frequency) approach, in which term weight is proportional to term frequency and inversely proportional to the number of documents in which a term occurs [8].[6]

Regarding the use case originally envisaged for this work, the next step is to refine the relationships building process to make it straightforward to determine relationships for documents as they are incrementally added to the corpus and indexed. Following that, the aim is to integrate related article functionality into the Windows phrase navigator. Further out, it would be interesting to investigate possibilities for developing the method to make it applicable more widely, to significantly larger corpora, paying close attention to compute time requirements and incremental operation. In addition, it is intriguing to think about what might be possible if some of the ideas outlined above were combined with more lexically informed and/or syntactically aware approaches. Once it becomes possible to determine, visualize and make effortlessly navigable the semantic relationships, indeed the key epistemic connections, which exist within the total academic content space, attention must inevitably focus on the role of the journal. So much of the existing scholarly publishing infrastructure, with all its costs and complexity, exists in order to persist into a digital future a way of organizing the academic content space that took shape during the age of the printing press. Work like that reported here, which promises to deliver new methods for apprehending and negotiating the structure of that content space, can be seen to pose searching questions about the role and status of journals [14].

References

1. Aryal, S., et al. (2019) A new simple and effective measure for bag-of-word inter-document similarity measurement. *arXiv*: 1902.03402v1.
2. Boyack, K.W., et al. (2011) Clustering More than Two Million Biomedical Publications: Comparing the Accuracies of Nine Text-Based Similarity Approaches. *PLoS ONE* **6**(3): e18029.
3. Fagan, J.L. (1987) Automatic Phrase Indexing for Document Retrieval: An Examination of Syntactic and Non-Syntactic Methods. *SIGIR '87: Proceedings of the 10th annual*

[6] TF-IDF is reported to be the term weighting scheme employed in a large majority of article recommender systems [13].

international ACM SIGIR conference on Research and development in information retrieval (November 1987), pp.91–101. (DOI: 10.1145/42005.42016)

4. Haarman, T., et al. (2019) Unsupervised Keyphrase Extraction for Web Pages. *Multimodal Technologies and Interaction* **3**, 58. (DOI: https://doi.org/10.3390/mti3030058)
5. Maron, M.E. and Kuhns, J.L. (1960) On Relevance, Probabilistic Indexing and Information Retrieval. *Journal of the Association for Computing Machinery* **7**(3): 216-244.
6. Niyigena, P., et al. (2016) Efficient Document Similarity Detection Using Weighted Phrase Indexing. *International Journal of Multimedia and Ubiquitous Engineering* **11**(5): 231-244.
7. Robertson, S.E. and Spärck Jones, K. (1976) Relevance Weighting of Search Terms. *Journal of the American Society for Information Science* **27**: 129-146.
8. Spärck Jones, K. (1972) A statistical interpretation of term specificity and it application in retrieval. *Journal of Documentation* **28**(1): 11-21.
9. Thompson, P. (2008) Looking back: On relevance, probabilistic indexing and information retrieval. *Information Processing and Management* **44**: 963-970.
10. Harris, Z. (1991) *A Theory of Language and Information*. Oxford: Oxford University Press.
11. King, A.M., et al. (2007) Applications of ultrasonography in the reproductive management of *Dux magnus gentis venteris saginati*. *The Veterinary Record* **160**(3): 94-96. (DOI: 10.1136/vr.160.3.94)
12. Stagnaro, S. and Caramel, S. (2013) The key role of vasa vasorum inherited remodeling in QBS microcirculatory theory of atherosclerosis. *Frontiers in Genetics* **4**: article 55. (doi: 10.3389/fgene.2013.00055)
13. Beel, J., et al. (2016) Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries* **17**(4): 305–338.
14. Powell, A. (2012) Open scientific communication and peer review: speculations and prognostications. *Epistemic Systems* blog. https://epistemicsystems.wordpress.com/2012/03/03/open-scientific-communication-and-peer-review-speculations-and-prognostications/

Supplementary Materials

A. File relationships data for test corpus (.xlsx file)

External file: Powell_corpus-similarity-data.xlsx

B. Phrases indexed (using 4-2 indexing scheme as described), with corpus frequencies and host file counts (.xlsx file)

External file: Powell_phrases.xslx

C. Appendices (PDF file)

External file: Powell_Appendices.pdf